Universidad Autónoma de Madrid
Departamento de Ingeniería Informática

Reglas Genéricas

Objetos, Reglas y Excepciones en el Procesamiento del Lenguaje Natural

Memoria de Tesis Doctoral
presentada por

Julio Gonzalo Arroyo

Dirigida por

Pilar Rodríguez Marín

Junio 1995

A mis padres, Bernardo y Mari Luz, que lo pasaron casi tan mal como yo cuando no quería ir al colegio; que lo pasaron peor que yo cuando me examiné de selectividad; y que es como si hubieran hecho esta memoria letra por letra sin necesidad de haberla visto.

Agradecimientos

Estoy en deuda con muchas personas. La primera es Roberto Moriyón, que será para siempre mi maestro. La segunda es Pilar Rodríguez, que me conoce tan bien como mi madre, y sin la cual esta travesía difícilmente hubiera llegado a buen término. La tercera es Eduard Hovy, que en las tres semanas que trabajamos juntos me llenó de ideas y energías para varios años. Desde entonces no ha cesado de animarme y ayudarme. Estoy también en deuda con Felisa Verdejo. En el poco tiempo que ha pasado desde que nos conocimos ya me ha enseñado - quizás sin darse cuenta - algo fundamental: que el movimiento se demuestra andando.

Además de ellos, varias personas han influido de manera decisiva con sus críticas y sugerencias en el curso que ha tomado mi trabajo. Ted Briscoe me hizo valoraciones y sugerencias que no tienen precio. De Horacio Rodríguez he recibido también valiosos comentarios técnicos. En Teresa Solías he encontrado a la lingüísta aguda y sin prejuicios que suple mis carencias; su colaboración en el trabajo que se expone en el capítulo 7 de esta memoria ha sido decisiva. Finalmente, las charlas que mantuve con Holger Maier y Stuart Shieber durante la VI ESSLLI me han sido también de gran ayuda, así como los comentarios de Chelo Rodríguez.

En otro orden de cosas, Nacho Mayorga y Miguel Rodríguez me han descargado de tanto trabajo mientras estaba en la fase de redacción que tardaré mucho tiempo en poder compensarles.

Gracias, como no , al IIC por acogerme y por todos los medios técnicos que puso a mi alcance.

Gracias a Pablo Castells, Javier Contreras, Paco Saiz, Julia Díaz, José María Villanueva y a toda la tropa del despacho quince por todos estos años estupendos. Muchas gracias a mis hermanos y a todos mis amigos por su aliento en los momentos difíciles, que gracias a ellos ni siquiera se pueden llamar difíciles. Gracias en especial a Dani y a Fernando, que han inundado mi correo de palabras reconstituyentes en las últimas semanas.

Y por último, gracias a ti, Eva, por mover mi sangre más que mi propio corazón durante todo este tiempo.

Resumen

Presentamos un formalismo metagramatical - el de las *reglas genéricas* - para expresar y dar una interpretación por defecto a las reglas gramaticales en los sistemas de procesamiento de lenguaje natural.

Inspirándose en los principios de la programación orientada a objetos, nuestro formalismo introduce un mecanismo de *ligadura dinámica* entre el nivel de representación lingüística y el nivel de parsing. Tanto el comportamiento genérico como el excepcional son interpretados como funciones orientadas a objeto, organizadas de acuerdo con un orden parcial que establece la precedencia entre ellas. Este enfoque permite un desarrollo modular e incremental de la gramática, reflejando de forma directa la diferencia entre comportamiento regular y excepcional.

Como aplicación teórica en el campo de la sintaxis, presentamos un reformulación de un tratamiento para la coordinación de no constituyentes dentro de las gramáticas categoriales. Mientras que la gramática original es computacionalmente intratable y no captura el carácter excepcional de las reglas asociadas con ese fenómeno, su reformulación como regla genérica tiene unas propiedades computacionales equiparables a las de una gramática libre de contexto, y reduce drásticamente el espacio de búsqueda asociado al proceso de parsing.

Como aplicación práctica, mostramos la utilización del formalismo de reglas genéricas en la concepción y desarrollo de un intérprete semántico de problemas de matemáticas enunciados en castellano, en el entorno del sistema PRÓGENES . Las reglas genéricas se revelan como un entorno de representación muy adecuado para expresar formalismos de interpretación semántica inspirados en el cálculo λ, así como para regular su interacción con la sintaxis.

Abstract

We present a metagrammatical formalism, *generic rules*, to give a default interpretation to grammar rules.

Drawing from object-oriented programming concepts, our formalism introduces a *dynamic binding* mechanism interfacing the level of pure grammatical knowledge representation and the level of parsing. Both generic and specific behaviour are interpreted as object-oriented functions that map daughter information into mother features, organized as a partial order that induces precedence between them. Such approach provides modular and incremental grammar writing, explicitly reflecting the difference between regular and exceptional behaviour.

As a syntactic case study, we present an approach to non-constituent coordination within categorial grammars, and reformulate it as a generic rule. While the original grammar is computationally intractable and does not capture the exceptional sense of the rules associated to non-constituent phenomena, the generic rule is context-free parsable and provides a significant reduction of the search space associated to the parsing task.

As a practical application, we describe the conception and development of a semantic interpreter of mathematical problems within a generic rules approach, in the framework of the PRÓGENES project. Generic rules are a suitable environment to represent semantic interpretation approaches based on λ-calculus, and to express a modular syntax-semantics interface.

Contenidos

Capítulo 1

Introducción

¿Cómo se puede describir e interpretar computacionalmente de forma óptima la interacción entre el comportamiento regular y excepcional en un lenguaje?

Las teorías gramaticales siempre hacen distinción entre fenómenos regulares - o *no marcados* - y fenómenos excepcionales o *marcados*. Un conjunto de reglas o mecanismos básicos da cuenta de los primeros, mientras que los casos excepcionales requieren a menudo la introducción de mecanismos más complejos y computacionalmente más costosos. Esto es particularmente cierto en las aproximaciones lexicalistas al lenguaje, en las que la mayor parte de la información se deposita en los objetos lingüísticos, mientras que el conjunto de reglas que relacionan esos objetos se reduce a unos pocos mecanismos, muy generales, de combinación de información.

El fenómeno de la coordinación de no constituyentes es un buen ejemplo. En general, una conjunción combina dos sintagmas. Existen casos, sin embargo, en los que la conjunción relaciona elementos lingüísticos que no forman un sintagma. Por ejemplo, en la oración

Eva soñó con la nieve en Santa Cruz y con el mar en Madrid.

la conjunción *y* está coordinando *con el mar en Madrid* y *con la nieve en Santa Cruz*, que no son sintagmas. Sería de esperar que la aplicación de reglas capaces de formar falsos sintagmas de este tipo tuvieran un carácter excepcional dentro de la gramática que diera cuenta de ellos, licenciándose su aplicación sólo en las situaciones específicas que lo requirieran. Sin embargo, no es éste el caso. Como veremos a lo largo de esta memoria, el tratamiento de la coordinación de no constituyentes se hace mediante reglas que modifican por completo las propiedades computacionales de la gramática base, disparando el coste computacional de los procesos de parsing. A cambio, se introducen todo tipo de heurísticas ad-hoc en los algoritmos de parsing para podar el espacio de búsqueda y minimizar los efectos de la explosión combinatoria que producen este tipo de reglas.

El origen de la dificultad para expresar formalmente la diferencia entre reglas por defecto y reglas excepcionales se encuentra en la forma de asignar estructura a las oraciones en los formalismos gramaticales, desde las gramáticas libres de contexto (context-free grammars o CFGs) hasta las estructuras de rasgos tipadas. Concretemos mediante un sencillo ejemplo.

Consideremos una gramática libre de contexto - posiblemente aumentada mediante restricciones funcionales asociadas a cada regla - que incluye la siguiente jerarquía:

$$\mathsf{VP} \rightarrow \mathsf{VP}_1 \mid \mathsf{VP}_2 \mid \mathsf{VP}_3 \mid \mathsf{VP}_4 \mid \mathsf{VP}_i \tag{1.1}$$

y en la que disponemos de la regla

$$\mathsf{S} \rightarrow \mathsf{NP\ VP} \tag{1.2}$$

que licencia la construcción de una oración a partir de un predicado nominal y un predicado verbal. Al desarrollar la gramática encontramos que la combinación de cierto tipo de predicados verbales VP_i no se ajusta a esta regla, y produce un tipo especial de oración S_i, donde $\mathsf{S} \rightarrow \mathsf{S}_i$ [1]. La forma más natural de expresar este hecho sería mediante una nueva regla que da cuenta de ese comportamiento excepcional:

$$\mathsf{S}_i \rightarrow \mathsf{NP\ VP}_i \tag{1.3}$$

Sin embargo, esta regla no captura el carácter excepcional de este fenómeno. Es decir, esta regla no expresa que la combinación de un NP con un objeto de tipo específico VP_i debe hacerse a través de la regla 1.3 y *no a través de la regla 1.2.* Una gramática libre de contexto con esas dos reglas mantendrá las dos posibilidades. Será posible la derivación que ibamos buscando:

$$\mathsf{S} \Rightarrow \mathsf{S}_i \Rightarrow \mathsf{NP\ VP}_i \tag{1.4}$$

pero también esta otra, no deseada:

$$\mathsf{S} \Rightarrow \mathsf{NP\ VP} \Rightarrow \mathsf{NP\ VP}_i \tag{1.5}$$

Para que la regla excepcional funcione como tal no hay mas remedio que reconsiderar el resto de la gramática. Una posibilidad es replantearnos el conjunto de reglas, sustituyendo la regla 1.3 por las siguientes:

[1] Este podría ser el caso de oraciones como "Eva lee un libro" (caso regular) frente a "Eva es increible" (caso excepcional). Los predicados verbales copulativos o predicativos influyen en la asignación de funciones para las oraciones de las que forman parte.

$$S := NP\ VP_1 \mid NP\ VP_2 \mid NP\ VP_3 \mid NP\ VP_4 \tag{1.6}$$

Otra posibilidad es intentar mantener constante el número de reglas binarias, reescribiendo tanto la jerarquía como las reglas:

$$\begin{aligned} VP &:= VP_i \mid VP_t \\ VP_t &:= VP_1 \mid VP_2 \mid VP_3 \mid VP_4 \\ S &:= NP\ VP_t \\ S_i &:= NP\ VP_i \end{aligned} \tag{1.7}$$

Ninguna de estas posibilidades captura el carácter excepcional de la regla 1.3. En otras palabras, una gramática no puede ser desarrollada incrementalmente de esta forma, ya que el introducir comportamientos específicos conflictivos con las especificaciones generales nos obliga a revisar todos los tipos léxicos y las reglas introducidas previamente.

Este caso trivial ejemplifica un problema de representación que se vuelve muy significativo cuando una operación o regla realiza la mayor parte de las operaciones de composición de información lingüística. Esta situación es, precisamente, la que se da en la mayoría de los formalismos gramaticales y computacionales actuales, en los que se dispone de entradas léxicas muy ricas en información que se relacionan entre ellas mediante un número muy reducido de mecanismos o reglas para combinar esa información. En el caso de las gramáticas categoriales, por ejemplo, la gramática básica se reduce a dos reglas direccionales de aplicación funcional. La posibilidad de establecer excepciones a esta regla, como en el caso de la coordinación de no constituyentes, puede ser crucial para la adecuación computacional del sistema. La misma situación se produce en las aproximaciones a la interpretación semántica basadas en la lógica y el cálculo λ, en las que los mecanismos de abstracción λ y conversión β llevan a cabo la mayor parte de los procesos de composición semántica.

Sin embargo, todos estos formalismos se comportan de la misma forma que las gramáticas libres de contexto frente al problema de representación que acabamos de presentar. Desde los sistemas de reglas libres de contexto aumentadas con restricciones de Unificación hasta los sistemas de estructuras de rasgos tipadas con herencia por defecto, la posibilidad de establecer reglas sintagmáticas genéricas y excepcionales no ha sido contemplada.

Sin embargo, en los últimos años se ha estudiado en profundidad la posibilidad de expresar información léxica por defecto en sofisticados sistemas de herencia, mostrando un paralelismo claro con la representación de los datos en los lenguajes de programación orientados a objetos. Un posible motivo de esta descompensación es que los sistemas de estructuras de rasgos permiten expresar la información sintagmática como parte de las restricciones funcionales sobre los objetos del dominio, haciendo que se desvanezca la diferencia entre una regla libre de contexto y sus restricciones asociadas. La posibilidad de representar toda la información lingüística

mediante una jerarquía de estructuras de rasgos, y de interpretar esta jerarquía mediante operaciones no monótonas (sustituyendo a la Unificación), parece abrir la puerta a la representación de reglas excepcionales. En la primera parte de la tesis discutiremos esta posibilidad, viendo que no es éste el caso.

Esta memoria está dedicada a presentar un formalismo que permite expresar directamente reglas genéricas y excepcionales que relacionen cualquier tipo de información lingüística, sintáctica o semántica. La idea esencial de las *reglas genéricas*, como llamaremos a este formalismo, es la de introducir un proceso de *ligadura dinámica* entre los procesos de parsing, por un lado, y las reglas de la gramática por otro. Este proceso está inspirado en los *métodos* de los lenguajes de programación orientada a objetos. Cuando un algoritmo de parsing invoca la gramática para combinar un par de objetos lingüísticos, el proceso de ligadura dinámica establece un orden de precedencia entre las reglas aplicables, regulando su aplicación a partir de este ordenamiento.

Este enfoque tiene varias ventajas, como veremos. Por un lado, admite los algoritmos de parsing que se aplican sobre gramáticas libres de contexto, manteniendo las propiedades computacionales de esos algoritmos casi intactas. Por otro lado, apenas restringe el tipo de teoría en el que se fundamentan las reglas y los objetos lingüísticos. Proporciona, además, una nueva visión de la interfaz entre sintaxis y semántica, permitiendo el desacoplamiento de reglas sintácticas y semánticas y una interacción flexible y modular entre ambas.

En la primera parte de la memoria veremos la diferencia entre la información sobre la estructura de las oraciones - cuyo representante canónico son las gramáticas libres de contexto - y transmisión de información a través de los nodos de esa estructura mediante operaciones como la Unificación. Veremos que ninguno de los recursos expresivos desarrollados para uno y otro tipo de información permiten expresar reglas por defecto.

En la segunda parte, un repaso a la forma de escribir programas asociados a una jerarquía de datos en programación orientada a objetos nos sugerirá una forma de obtener comportamiento por defecto de las reglas gramaticales. Desarrollaremos un formalismo que adopta los mecanismos clave de la programación orientada a objetos sin heredar el carácter procedural de éstos. A pesar de que nuestro sistema de representación introduce comportamientos no monótonos, sus propiedades computacionales son parecidas a las de las gramáticas libres de contexto.

Formularemos también un algoritmo de reescritura que nos permitirá pasar de un sistema de reglas por defecto a un sistema monótono de reglas sintagmáticas. Este algoritmo permite establecer sólidamente la relación de las reglas genéricas con las gramáticas sintagmáticas usuales.

Para terminar, presentaremos dos aplicaciones del formalismo. La primera, de carácter teórico, ofrece una solución al problema de la coordinación de no-constituyentes al nivel de descripción gramatical (sin postular niveles metagramaticales ni introducir heurísticas de parsing) que reduce el espacio de búsqueda

tanto como las aproximaciones procedurales basadas en heurísticas. Ese trabajo es fruto de una colaboración con Teresa Solías.

La segunda, de carácter práctico, muestra la utilización del formalismo de reglas genéricas en la concepción y desarrollo de un intérprete semántico de problemas de matemáticas enunciados en castellano, en el entorno del sistema PRÓGENES . Las reglas genéricas se revelan como un entorno de representación muy adecuado para expresar formalismos de interpretación semántica inspirados en el cálculo λ, así como para regular su interacción con la sintaxis.

A lo largo del texto se han traducido al castellano algunos términos técnicos, mientras que para otros se ha dejado la expresión original en inglés. El criterio para esta distinción es el de reservar el término original para aquellos conceptos cuyo equivalente en castellano es poco utilizado o pierde parte de las connotaciones de la palabra original, como en el caso de "parsing" y "análisis".

Parte I

Planteamiento del problema: representación del comportamiento excepcional

Capítulo 2

Elementos básicos de representación gramatical

¿ Qué capacidad expresiva, y qué propiedades computacionales, tienen los sistemas de representación de conocimiento lingüístico actuales ? Cuáles son sus limitaciones para expresar comportamiento regular y excepcional ?

En éste capítulo estudiaremos los dos recursos básicos de descripción formal de un lenguaje dentro del paradigma de estructura sintagmática. Por un lado, las gramáticas libres de contexto, como mecanismo fundamental para dar cuenta de la estructura de las oraciones. Por el otro, las estructuras de rasgos como dominio informacional de objetos lingüísticos, y la Unificación como mecanismo de combinación de esa información a través de los nodos de un análisis. Ninguno de los dos, ni las posibles interacciones entre ambos, nos permiten expresar reglas por defecto.

2.1 Gramáticas libres de contexto

Las gramáticas formales dan cuenta de la estructura de un lenguaje por medio de un conjunto de símbolos terminales (el léxico), de un conjunto de símbolos no-terminales (que etiquetan los nodos de las derivaciones) y de un conjunto de reglas de producción que asocian terminales y no terminales. Atendiendo a la forma de esas reglas, Chomsky definió una jerarquía de cuatro clases de gramáticas como modelos potenciales para las descripción de las lenguas naturales [Chomsky, 1956, Chomsky, 1959]. Las gramáticas libres de contexto [1] son aquellas en las que la parte izquierda de cada regla de producción consiste en un único símbolo no terminal. Recordemos su definición.

[1] Utilizaremos el término "gramáticas libres de contexto", más usado, en lugar de "gramáticas independientes del contexto", traducción más adecuada del término inglés *context-free grammar*, pero menos utilizado.

Una *cadena* es una secuencia finita de símbolos. La longitud de una cadena es el número de símbolos que la componen. La cadena vacía ϵes aquella cadena con longitud cero (es decir, sin ningún símbolo). La *concatenación* de dos cadenas es la cadena formada por la primera seguida de la segunda. Un *alfabeto* es un conjunto finito de símbolos. un *lenguaje formal* es un conjunto de cadenas de símbolos pertenecientes a un mismo alfabeto. Dado un alfabeto Σ, representamos el lenguaje compuesto por todas las cadenas posibles sobre ese alfabeto como Σ^*.

2.1.1 Definición de gramática libre de contexto

Una *gramática libre de contexto* es una tupla $G = \langle N, \Sigma, P, S \rangle$, donde

- N es un conjunto de símbolos llamados *no terminales.*
- S es un elemento distinguido de N llamado *símbolo de partida.*
- Σ es un conjunto de símbolos llamados *terminales*, y $V = N \cup \Sigma$ es el *vocabulario* de la gramática.
- P es un conjunto de producciones de la forma $A \rightarrow \alpha$, donde $A \in N$ y $\alpha \in V^*$.

Usaremos el símbolo $\Rightarrow$ para representar la *derivación inmediata*: si $A \rightarrow \beta$ es una producción de P y α y γ son cadenas de V^*, entonces $\alpha A \gamma \Rightarrow \alpha\beta\gamma$. Decimos que $\alpha\beta\gamma$ deriva de forma inmediata de $\alpha A\gamma$.

Usaremos el símbolo $\overset{*}{\Rightarrow}$ para la clausura reflexiva y transitiva de la relación $\Rightarrow$. Es decir, si $\alpha_1, \alpha_2, \ldots, \alpha_m$ son cadenas en V^* y

$$\alpha_1 \Rightarrow \alpha_2, \alpha_2 \Rightarrow \alpha_3, \ldots, \alpha_{m-1} \Rightarrow \alpha_m \tag{2.1}$$

entonces decimos que $\alpha_1 \overset{*}{\Rightarrow} \alpha_m$, o que α_m se deriva de α_1 para esa gramática.

Se llama *Lenguaje generado por G* al conjunto

$$L(G) \equiv \{w \mid w \in \Sigma^* \text{ y } S \overset{*}{\Rightarrow} w\} \tag{2.2}$$

El nombre de "gramáticas libres de contexto" viene del hecho de que una regla de producción puede aplicarse sobre un símbolo independientemente de la cadena

en que esté incluido. Es decir, si disponemos de una regla $A \rightarrow w$, podemos hacer la derivación $\alpha A \beta \Rightarrow \alpha w \beta$ independientemente de quienes sean α y β.

Las gramáticas libres de contexto pueden ser reconocidas en tiempo cúbico en el tamaño del problema (en nuestro caso, la longitud de la cadena). Nos interesa poner de manifiesto esta dependencia cúbica, para poder comparar los procesos de parsing con los del formalismo de reglas genéricas, una vez que lo hayamos introducido. Utilizaremos para ello el algoritmo Cocke-Younger-Kasami (CYK) [Younger, 1967, Kasami, 1965] para reconocer gramáticas libres de contexto en forma normal de Chomsky, sobre el que es fácil ver la dependencia n^3.

Presentaremos el algoritmo de una forma algo diferente de la usual en la literatura sobre el tema, para hacerlo más intuitivo.

Sea una gramática libre de contexto $G = \langle N, \Sigma, P, S \rangle$ y una cadena $w_1 \ldots w_n$ en Σ. Consideramos una matriz triangular A de tamaño n^2 asociada a esa cadena . Definimos los elementos de la matriz como sigue:

$$
\begin{aligned}
&A_{ii} = \{w_i\}, i \leq n \\
&A_{ij} = \overline{v}_{ij} \bullet \overline{w}_{ij}, i < j \text{ donde} \\
&\overline{v}_{ij} = (A_{ii} A_{ii+1} \ldots A_{ij-1}) \\
&\overline{w}_{ij} = (A_{i+1j} A_{i+2j} \ldots A_{jj}) \\
&\overline{v}_{ij} \bullet \overline{w}_{ij} = \bigcup_{k=i \ldots j-i} G(v_{ij}^k, w_{ij}^k) \\
&G(a^1, a^2) = \begin{cases} \{B\} \text{ si } \exists B \mid B \rightarrow v^1 v^2 \in P \\ \emptyset \text{ en caso contrario} \end{cases}
\end{aligned}
\tag{2.3}
$$

El elemento de matriz A_{ij} representa el posible análisis parcial correspondiente a la subcadena $w_i \ldots w_j$. Es decir, $A_{ij} = B$ equivale a $B \stackrel{*}{\Rightarrow} w_i \ldots w_j$. En la figura 2.1 puede verse como se forma el elemento de matriz A_{14} mediante el *producto escalar* '$\bullet$' de los vectores que determina A_{14} a por su izquierda y por debajo.

La descripción de esta matriz constituye, en sí misma, el algoritmo CYK. Sólo es necesario rellenar las casillas de forma que no se asigne un valor a A_{ij} hasta que no tengan valor los elementos de cada uno de sus dos vectores asociados $\overline{v}_{ij}$ y $\overline{w}_{ij}$. La cadena queda reconocida una vez que se rellena la casilla A_{1n}, que corresponde a la categoría que la gramática asocia a la cadena entera $w_1 \ldots w_N$.

El número de casillas que se intentan rellenar es del orden de n^2. Cada casilla A_{ij} involucra realizar la operación G, que invoca la gramática, $j - i$ veces, que es

$$\Sigma = \{ \text{Alfredo , bichos, odia , los } \}$$

$$N = \{\mathsf{S}, \mathsf{NP}, \mathsf{VP}, \mathsf{V}_t, \mathsf{N}, \mathsf{Det}\}$$

$$P = \left\{ \begin{array}{l} \mathsf{S} \rightarrow \mathsf{NP\ VP} \\ \mathsf{VP} \rightarrow \mathsf{V}_t\ \mathsf{NP} \\ \mathsf{NP} \rightarrow \mathsf{Det\ N} \\ \mathsf{NP} \rightarrow \text{Alfredo} \\ \mathsf{N} \rightarrow \text{bichos} \\ \mathsf{Det} \rightarrow \text{los} \\ \mathsf{V}_t \rightarrow \text{odia} \end{array} \right\}$$

N : Alfredo	$\emptyset$: Alfredo odia	$\emptyset$: Alfredo odia los	S: Alfredo odia los bichos
	V_t: odia	$\emptyset$: odia los	VP: odia los bichos
		Det: los	NP: los bichos
			N: bichos

$$A_{14} = \overline{v}_{14} \bullet \overline{w}_{14} = (\mathsf{N}\ \emptyset\ \emptyset) \bullet (\mathsf{VP\ NP\ N}) = G(\mathsf{N}, \mathsf{VP}) \cup G(\emptyset, \mathsf{NP}) \cup G(\emptyset, \mathsf{N}) = \{\mathsf{S}\}$$

Figura 2.1: Análisis de la cadena "Alfredo odia los bichos" según el algoritmo CYK en nuestra presentación particular. Remarcamos con doble línea el elemento A_{14} y los dos vectores que originan ese análisis parcial,$\overline{v}_{14}$y $\overline{w}_{14}$

también del orden de n. De forma que, en total, la operación básica de combinar dos categorías se realiza en un tiempo $\mathcal{O}(n^3)$. Si la gramática es ambigua y queremos dar cuenta de todos los árboles posibles de análisis, hay que desglosar la operación $G(A_{ij}, A_{kl})$ en una operación de combinación por cada par de análisis posibles correspondientes a A_{ij} y a A_{kl}. En ese caso, la complejidad es exponencial. En la práctica, los sistemas tienen un límite para el número de análisis ambiguos de un constituyente, de forma que el coste con el tamaño de la cadena se mantiene como n^3 [Carroll, 1994]. Hay otra variable interesante respecto a la complejidad de los procesos de parsing, que es la longitud de la cadena. En el caso del algoritmo CYK tal y como lo hemos formulado, la única dependencia con el tamaño de la gramática se da al invocar la gramática mediante la función G. En el peor de los casos, establecer si existe una regla B $\rightarrow$ C D para combinar dos símbolos no terminales C D requiere revisar todas las reglas en P. Por lo tanto, el tiempo de reconocimiento es un $\mathcal{O}(|G|)$.

Respecto a la forma en que se representa el conocimiento, este tipo de gramáticas vierten dos tipos de información diferente, la información sobre los objetos lingüísticos y sobre las relaciones entre ellos, en un solo mecanismo formal: el conjunto de reglas de producción. Efectivamente, el léxico está representado por un simple conjunto de símbolos; ni el conjunto está dotado de ninguna estructura, ni cada símbolo individual es más que una cadena de caracteres. La información sobre la categoría de cada palabra se especifica como reglas unarias con los terminales en la parte derecha.

Los formalismos que se utilizan actualmente para describir y procesar lenguajes naturales exceden con mucho la capacidad generativa de las gramáticas libres de contexto y sus propiedades computacionales. Sin embargo, la información suele distribuirse en torno a un esqueleto sintáctico, más o menos explícito, en forma de reglas libres de contexto. El procesamiento de este tipo de marcadores sintácticos se hace mediante algoritmos y técnicas directamente relacionados con las gramáticas libres de contexto.

2.2 Restricciones funcionales y Unificación

Los mecanismos inherentes a las gramáticas libres de contexto nos proporcionan un punto de partida para expresar la sintaxis de un lenguaje de forma declarativa y utilizarla en los procesos de búsqueda relacionados con el problema de parsing. Naturalmente, una gramática libre de contexto en sí misma no resuelve ninguno de los problemas en Inteligencia Artificial relacionados con la comprensión del lenguaje. La estructura de los sintagmas es un punto de partida para extraer del lenguaje natural la información que necesitemos (para servir de interfaz con una base de datos, para traducción automática, para dar una interpretación semántica..), a partir de un enfoque composicional que nos permita controlar el flujo de información a través de los nodos del análisis.

Los primeros sistemas de procesamiento de lenguaje natural consistían usualmente en gramáticas libres de contexto aumentadas mediante operaciones asociadas de combinación de información. Estas operaciones no estaban formalizadas sistemáticamente, y consistían usualmente en código arbitrario. Las *Redes de transición aumentadas* permitieron, en un segundo estadio, escribir gramáticas más complejas de una forma más o menos uniforme, y dieron lugar a una gran cantidad de trabajo de investigación en el campo del procesamiento del lenguaje natural. Poco a poco, la preocupación de lingüistas e informáticos por las propiedades formales y computacionales del dominio informacional adecuado para representar el conocimiento sobre las lenguas naturales llevó a una vuelta a las gramáticas sintagmáticas aumentadas. "Aumentadas", en este caso, tiene un sentido diferente al que se le dio en las etapas iniciales de la investigación sobre el procesamiento del lenguaje. Se trata de las llamadas *Gramáticas de Unificación* o *Gramáticas basadas en restricciones* (Constraint Based Grammars) [Kay, 1983, Shieber, 1986, Shieber, 1992]. Los objetos lingüísticos pertenen a un dominio informacional bien definido, el de las *estructuras de rasgos* (feature structures). El flujo de información se estructura por medio de la operación de Unificación. Finalmente, la única operación permitida entre cadenas es la concatenación: este hecho relaciona directamente a las gramáticas de unificación con las gramáticas libres de contexto, aunque su poder generativo y sus propiedades computacionales sean bastante diferentes.

2.2.1 Estructuras de rasgos

Una estructura de rasgos es un conjunto de pares atributo-valor. El valor de un atributo puede ser un símbolo atómico o ser, a su vez, una estructura de rasgos. Dos atributos de una estructura de rasgos pueden compartir un mismo valor, de la misma forma en que dos variables pueden apuntar al mismo registro en memoria. En la figura 2.2 pueden verse un par de ejemplos.

La primera estructura de rasgos de la figura 2.2 podría corresponder a la descripción del objeto lingüístico asociado al sintagma "Alfredo". El atributo *fon* da la cadena asociada a este sintagma, y el atributo *concordancia* nos da los rasgos flexivos del sintagma. La segunda estructura podría corresponder a una representación sencilla de la oración "Alfredo escucha". Los índices representan compartición de variables. Se dice que una estructura de rasgos es *reentrante* cuando algún par de atributos de esa estructura comparten un valor común. Mediante ellos representamos que los valores de los atributos *numero* y *persona* del NP en función de sujeto y del verbo son idénticos. De la misma forma, representamos que la cadena correspondiente a la oración se forma a partir de la concatenación de la cadena correspondiente al NP y la cadena correspondiente al VP. La notación $< \cdots >$ para listas es una abreviatura de la definición de una lista como una estructura de rasgos con dos atributos *primero* y *resto*, como en la definición

$$\begin{bmatrix} fon & 'Alfredo' & \\ cat & NP & \\ concordancia & \begin{bmatrix} numero & singular \\ persona & tercera \\ genero & masculino \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} fon & \langle 'Alfredo\ \boxed{1}\ ,\ 'escucha'\ \boxed{2} \rangle & & \\ cat & S & & \\ hijos & \left\langle \begin{bmatrix} fon & \boxed{1} & & \\ cat & NP & & \\ concordancia & \begin{bmatrix} numero & singular\ \boxed{3} \\ persona & tercera\ \boxed{4} \\ genero & masculino \end{bmatrix} \end{bmatrix}, \begin{bmatrix} fon: & \boxed{2} & & \\ cat & VP & & \\ subject & \begin{bmatrix} concordancia & \begin{bmatrix} numero & \boxed{3} \\ persona & \boxed{4} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figura 2.2: Dos objetos lingüísticos representados como estructuras de rasgos.Se usa la notación $\boxed{1}$ para indicar compartición de valores entre dos rasgos.

recursiva de lista.

El concepto de estructura de rasgos puede formalizarse de varias maneras. Por ejemplo, pueden verse como grafos dirigidos acíclicos con un nodo raiz (*rooted, directed, acyclic graph structures* o DAGs) [Gazdar and Mellish, 1989, Shieber, 1986]. Aquí vamos a recoger la formalización de [Kasper and Rounds, 1986, Kasper and Rounds, 1990] en términos de autómatas finitos etiquetados:

Dado un conjunto finito de atributos, Atributos, y un conjunto finito de átomos Atomos, una *estructura de rasgos* es una tupla $F = \langle Q, q_0, \delta, \alpha \rangle$ donde:

- Q es un conjunto finito de nodos.
- $q_0 \in Q$ es el nodo raiz.
- δ : Atributos $\times Q \rightarrow Q$ es la función de transición o función parcial de atributos en valores.
- $\alpha : Q \rightarrow$ Atomos es la función parcial de atributos en valores atómicos.

con las siguientes restricciones:

Conectividad Debe haber algún camino desde la raiz hasta cada uno de los nodos.
Los caminos se definen como secuencias de cero o más atributos. Sea Caminos = Atributos* y ϵ el camino vacío. La definición de la función de transición puede ser extendida a los caminos tomando $\delta(\epsilon, q) = q$ y $\delta(f.\pi, q) = \delta(\pi, \delta(f, q))$. La condición de conectividad puede expresarse entonces como la restricción de que haya un camino π tal que $q = \delta(\pi, q_0)$ para todo $q \in Q$.

Valores atómicos Sólo los nodos sin atributos pueden ser valores atómicos, de forma que si $\alpha(q)$ está definido para un nodo q, entonces $\delta(f, q)$ no está definido para ningún $f \in$ Atributos.

Aciclicidad El grafo debe ser acíclico, en el sentido en que no hay un camino π y un camino no vacío π' tales que $\delta(\pi, q_0) = \delta(\pi.\pi', q_0)$.

2.2.2 Unificación

Sobre las estructuras de rasgos se define de forma natural un orden parcial de *subsunción*. Una estructura de rasgos R subsume a otra R' si la información que contiene R' es una extensión de la que contiene la primera. Por ejemplo, para las dos estructuras de rasgos:

$$\begin{bmatrix} numero & singular \\ genero & masculino \\ persona & tercera \end{bmatrix} \qquad \begin{bmatrix} numero & singular \\ genero & masculino \end{bmatrix}$$

La segunda estructura subsume a la primera, pues la primera contiene más información que aquella. Esta relación de orden es parcial, puesto que dos estructuras con información incompatible no guardan relación alguna de subsunción. Por ejemplo:

$$\begin{bmatrix} numero & singular \\ genero & masculino \\ persona & tercera \end{bmatrix} \qquad \begin{bmatrix} numero & singular \\ genero & femenino \end{bmatrix}$$

La noción de subsunción puede ser formalizada asociando a cada estructura de rasgos un par de conjuntos que determinan las equivalencias de camino y los valores atómicos asignados a los caminos. Sea F una estructura de rasgos, y sea $\equiv_F$ la relación de equivalencia inducida entre caminos por los valores reentrantes en F. Sea $\mathcal{P}_F$ la función parcial inducida por F que lleva caminos en F a valores atómicos. Estas funciones nos llevan a la definición de *Estructura de rasgos abstracta*:

Si $F = \langle Q, q_0, \delta, \alpha \rangle$ es una estructura de rasgos, entonces el par $\langle \mathcal{P}_F, \equiv_F \rangle$ es la *estructura de rasgos abstracta* correspondiente a F, donde $\equiv_F \subseteq \mathsf{Caminos} \times \mathsf{Caminos}$ y $\mathcal{P}_F : \mathsf{Caminos} \rightarrow \mathsf{Atomos}$ son tales que:

- $\pi \equiv_F \pi' \iff \delta(\pi, q_0) = \delta(\pi', q_0)$ (*Equivalencia de caminos*)
- $\mathcal{P}_F(\pi) = \sigma \iff \alpha(\delta(\pi, q_0)) = \sigma$ (*Valor de camino*)

La estructura abstracta es suficiente para determinar el contenido de información de la estructura de rasgos, y por lo tanto es suficiente para determinar si una estructura de rasgos subsume a otra o no:

F *subsume* a F', $F' \sqsubseteq F$ si y sólo si:

- $\pi \equiv_F \pi' \Longrightarrow \pi' \equiv_F \pi$
- $\mathcal{P}_F(\pi) = \sigma \Longrightarrow \mathcal{P}_{F'}(\pi) = \sigma$

Es decir, F subsume a F' si y sólo si cada elemento de información de F está contenido en F'. Se llama $\top$ a la estructura de rasgos que tiene un solo nodo sin ningún valor atómico asociado. $\top$ subsume a todas las demás estructuras de rasgos.

$$A: \begin{bmatrix} cat & NP \\ concordancia & \begin{bmatrix} genero & masculino \end{bmatrix} \end{bmatrix}$$

$$B: \begin{bmatrix} concordancia & \begin{bmatrix} genero & masculino \\ numero & singular \end{bmatrix} \end{bmatrix}$$

$$A \cup B: \begin{bmatrix} cat & NP \\ concordancia & \begin{bmatrix} genero & masculino \\ numero & singular \end{bmatrix} \end{bmatrix}$$

Figura 2.3: Unificación de dos estructuras de rasgos A y B

La Unificación es aquella operación que compone la información de dos estructuras de rasgos compatibles. Puede verse un ejemplo en la figura 2.3. Se puede definir de forma sencilla en términos de la operación de subsunción:

La *Unificación* $F \cup F'$ de dos estructuras de rasgos F y F' es la máxima cota inferior de F y F' en la colección de estructuras de rasgos ordenada por la relación de subsunción. Por lo tanto

$$F \cup F' = F'' \iff \begin{cases} F'' \sqsubseteq F \\ F'' \sqsubseteq F' \\ \forall F''' | F''' \sqsubseteq F, F''' \sqsubseteq F' \Longrightarrow F''' \sqsubseteq F'' \end{cases} \tag{2.4}$$

La Unificación es una operación monótona e independiente del orden. Los formalismos basados estrictamente en la operación de Unificación son totalmente declarativos. Estas propiedades la hacen especialmente adecuada para tareas de representación de conocimiento, a pesar de que las estructuras de rasgos y la operación de Unificación son herramientas de representación extremadamente generales.

2.2.3 Reglas en gramáticas de Unificación

Una gramática de Unificación prototípica consiste en un esqueleto de reglas libres de contexto enriquecido con un conjunto de especificaciones sobre los rasgos (pares atributo valor) de los símbolos gramaticales que aparecen en las re-

$$
\begin{aligned}
&X_0 \rightarrow X_1\, X_2 \\
&\quad \langle X_0\, cat \rangle = s \\
&\quad \langle X_1\, cat \rangle = np \\
&\quad \langle X_2\, cat \rangle = vp \\
&\quad \langle X_0\, head \rangle = \langle X_2\, head \rangle \\
&\quad \langle X_0\, head\, subject \rangle = \langle X_1\, head \rangle
\end{aligned}
$$

Figura 2.4: Una regla de PATR-II, herramienta prototípica para describir gramáticas de unificación.

glas libres de contexto y en el léxico asociado. Este es el caso de herramientas de descripción gramatical como PATR-II [Shieber et al., 1983] o las Definite Clause Grammars [Pereira and Warren, 1980, Pereira and Warren, 1983] basadas en programación lógica, y de formalismos lingüísticos como LFG [Bresnan, 1982] o GPSG [Gazdar et al., 1985]. La unificación puede usarse para expresar identidad y restricciones de concurrencia entre valores de atributos, y para resolver los problemas de *percolación* o transmisión de esos valores a través de los nodos de una estructura de análisis (ver [Pulman, 1989] para una discusión del papel de la unificación en estos procesos).

El enfoque más usual de las herramientas para describir gramáticas de Unificación es el de representar las reglas gramaticales como abstracciones de reglas libres de contexto con restricciones funcionales acerca de los rasgos de los objetos que intervienen en cada regla. En la figura 2.4 puede verse un ejemplo de regla en PATR-II.

La parte "libre de contexto" expresa que esas restricciones se dan entre tres sintagmas, y que la cadena correspondiente al sintagma en la parte izquierda de la regla es la concatenación de las cadenas correspondientes a la parte derecha de la regla. Las igualdades establecen las categorías de los tres elementos, y establecen restricciones para la aplicación de la regla entre los valores de ciertos atributos de los sintagmas (como estructuras de rasgos) involucradas. Esas igualdades involucran, en un proceso composicional, la unificación de los términos a izquierda y derecha, reemplazando las subestructuras por sus unificaciones en las tres estructuras de rasgos asociadas con la aplicación de una regla. Las anotaciones a la regla libre de contexto no son, por tanto, operaciones o procedimientos separados de distinta naturaleza: son enunciados declarativos e independientes del orden acerca de las condiciones de buena formación de un nodo de análisis.

En la figura 2.5 puede verse un ejemplo de combinación de dos estructuras de

$$\textit{"Alfredo"} : \begin{bmatrix} \textit{cat} & \textit{NP} \\ \textit{cabecera} & \begin{bmatrix} \textit{concordancia} & \begin{bmatrix} \textit{numero} & \textit{singular} \\ \textit{persona} & \textit{tercera} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\textit{"escucha"} : \begin{bmatrix} \textit{cat} & \textit{VP} \\ \textit{cabecera} & \begin{bmatrix} \textit{forma} & \textit{finita} \\ \textit{sujeto} & \begin{bmatrix} \textit{concordancia} & \begin{bmatrix} \textit{numero} & \textit{singular} \\ \textit{persona} & \textit{tercera} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

$$\Longrightarrow$$

$$\textit{"Alfredo escucha"} : \begin{bmatrix} \textit{cat} & \textit{S} \\ \textit{cabecera} & \begin{bmatrix} \textit{forma} & \textit{finita} \\ \textit{sujeto} & \begin{bmatrix} \textit{concordancia} & \begin{bmatrix} \textit{numero} & \textit{singular} \\ \textit{persona} & \textit{tercera} \end{bmatrix} \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

Figura 2.5: Formación de la oración "Alfredo escucha" a partir de la regla de PATR-II de la figura 2.4

rasgos a partir de la regla 2.4.

Los formalismos de unificación añaden una serie de restricciones funcionales a las reglas de una gramática libre de contexto dentro de un formalismo con muchas propiedades formales y computacionales deseables. El papel de los objetos lingüísticos - léxico y descripciones sintagmáticas - se vuelve mucho más rico y complejo. Mientras con las reglas libres de contexto la práctica totalidad de la información lingüística se codificaba en las reglas de la gramática, en los formalismos de unificación la mayor parte de la información reside en el léxico, y ésta es combinada a través de las restricciones estructurales de las reglas libres de contexto y de un único mecanismo de combinación de información: la unificación.

Los procesos de búsqueda asociados a las reglas, sin embargo, son los mismos que con una gramática libre de contexto. Pueden aplicarse los mismos algoritmos de parsing (cuyo coste es ahora producto de la interacción entre los procesos de búsqueda y los procesos de unificación). Respecto a la posibilidad de dar un sentido excepcional o por defecto a las reglas gramaticales, nada cambia. De hecho, el sistema de reglas libres de contexto ha sido aumentado con una operación monótona, mientras que las interpretaciones por defecto de la información lingüística han de ser necesariamente no monótonas.

$$\begin{bmatrix} fon & \langle \boxed{1}, \boxed{2} \rangle \\ cat & S \\ hijas & \left\langle \begin{bmatrix} fon & \boxed{1} \\ cat & NP \\ concordancia & \begin{bmatrix} numero & \boxed{3} \\ persona & \boxed{4} \end{bmatrix} \end{bmatrix}, \begin{bmatrix} fon: & \boxed{2} \\ cat & VP \\ subject & \begin{bmatrix} concordancia & \begin{bmatrix} numero & \boxed{3} \\ persona & \boxed{4} \end{bmatrix} \end{bmatrix} \end{bmatrix} \right\rangle \end{bmatrix}$$

Figura 2.6: Esta estructura de rasgos incluye como restricciones funcionales tanto la concordancia sujeto-verbo como la información estructural equivalente a la regla libre de contexto S → NP VP.

2.2.4 Reglas como restricciones funcionales

Como hemos dicho, las estructuras de rasgos son un mecanismo de representación de conocimiento muy potente. Las restricciones funcionales que se especifican mediante valores reentrantes en las estructuras de rasgos puede servir, de hecho, para codificar las restricciones estructurales que hasta ahora habiamos representado como reglas de una gramática libre de contexto [Kartunnen and Kay, 1985]. Este es el enfoque de formalismos de representación como *Functional Unification Grammar* (FUG) [Kay, 1985] , *Typed Feature Structures* (TFS) [Zajac, 1992] y formalismos lingüísticos como *Head-Phrase Structure Grammar* (HPSG) [Pollard and Sag, 1987, Pollard and Sag, 1994]. Por ejemplo, la estructura de rasgos correspondiente a la oración "Alfredo escucha" de la figura 2.2 podría haber sido obtenida a partir de una regla codificada mediante la estructura de rasgos de la figura 2.6.

De esta forma, en sistemas como FUG las estructuras de rasgos son la única forma de especificar información lingüística. Las relaciones entre objetos quedan incluidas dentro de la propia descripción de los objetos. Esto induce un problema de representación: las distintas restricciones funcionales de una regla libre de contexto han de ser satisfechas simultáneamente; pero las reglas de una gramática han de ser satisfechas *disyuntivamente*. Por ejemplo, si disponemos de varias reglas para formar un predicado verbal:

$$\begin{array}{l} \mathsf{VP} \rightarrow \mathsf{V}_i \\ \mathsf{VP} \rightarrow \mathsf{V}_t\,\mathsf{NP} \\ \mathsf{VP} \rightarrow \mathsf{VP\,PP} \\ \ldots \end{array} \tag{2.5}$$

Estas reglas tendrán su equivalente en varias estructuras de rasgos que describen a una estructura de categoría VP. Pero un predicado verbal bien formado no tiene porque satisfacer cada una de esas estructuras, sino tan sólo una de ellas. Es en ese sentido en el que una gramática es disyuntiva. Para expresarlo, es necesario introducir un operador de disyunción entre estructuras de rasgos y extender la operación de unificación para considerarlo. En este caso, el proceso computacional de búsqueda tiene su origen precisamente en este operador de disyunción. Las tres reglas en la ecuación 2.5, por ejemplo, podrían tener su equivalente en la disyunción de las tres estructuras de rasgos que puede verse en la figura 2.7. Sin embargo, la introducción de un operador disyunción en un formalismo de estructuras de rasgos es un mecanismo de representación mucho más general y potente que una gramática libre de contexto, y tiene asociado un coste computacional mucho más alto (ver [Barton Jr. et al., 1987]).

$$\left[\begin{array}{ll} cat & VP \\ hijos & \left\langle \left[cat \quad Vt\right], \left[cat \quad NP\right] \right\rangle \end{array}\right] \vee \left[\begin{array}{ll} cat & VP \\ hijos & \left\langle \left[cat \quad Vi\right] \right\rangle \end{array}\right] \vee \left[\begin{array}{ll} cat & VP \\ hijos & \left\langle \left[cat \quad VP\right], \left[cat \quad PP\right] \right\rangle \end{array}\right]$$

Figura 2.7: Reglas como disyunción de estructuras de rasgos

En la práctica, la expresión del esqueleto libre de contexto como restricciones funcionales indistinguibles del resto de la información lingüística no trae demasiadas ventajas. Desde un punto de vista computacional, estos dos tipos de información son diferentes y llevan asociados mecanismos de resolución bien distintos. Expresar las reglas de la gramática como parte de las estructuras de rasgos nos impide hacer uso de las propiedades computacionales asociadas con las gramáticas libres de contexto y los algoritmos de parsing y sistemas desarrollados para tratarlas. De esta forma perdemos eficiencia y oscurecemos la diferencia entre procesos computacionalmente diferentes. Mientras que la tendencia de los linguistas ha sido la de utilizar este tipo de representación (como en HPSG), los sistemas de representación lingüística de bajo nivel optan usualmente por establecer una diferencia entre los mecanismos de combinación de información y los mecanismos de búsqueda asociados a la estructura, separando ambos tipos de procesos (ver [Copestake, 1992] para una discusión de este problema).

Cuando esta distinción no se establece a priori, se suelen utilizar mecanismos de reescritura que pasan de descripciones puramente funcionales a otras con el esqueleto libre de contexto desglosado del resto de la información (ver, por ejemplo, [Carroll, 1994, Gotz and Meurers, 1995]).

Sin embargo, la capacidad de expresar reglas como estructuras de rasgos parece abrir un camino hacia la representación de reglas por defecto. Efectivamente, la relación de subsunción establece un orden parcial entre estructuras de rasgos, y un orden parcial es la mejor forma de establecer precedencia entre distintos elementos informacionales. Si, en lugar de la Unificación, que es una operación monótona, encontraramos una extensión no monótona que se comportara como aquella donde no hubiera conflictos de información, y de forma distinta para resolver esos conflictos, podríamos pensar en la posibilidad de establecer reglas por defecto. En el capítulo siguiente estudiamos esa posibilidad, que no nos llevará, a pesar de todo, a la representación de reglas por defecto. La razón hay que buscarla en la distinta naturaleza de las restricciones estructurales (cuyo funcionamiento queremos cambiar) y las restricciones funcionales, sobre las que si pueden establecerse mecanismos de herencia por defecto.

Capítulo 3

Comportamiento excepcional versus información excepcional

3.1 Generalizaciones y herencia

Los conceptos de unificación y subsunción pueden ser utilizados para estructurar el dominio de objetos lingüísticos mediante una jerarquía de tipos. La jerarquización de los datos y el uso de la *herencia* es una técnica muy potente y ya habitual como sistema de representación de conocimiento y teoría computacional [Ait-Kaci, 1984, Cardelli, 1984], así como filosofía de programación [Meyer, 1988, Booch, 1990]. Una breve introducción al uso de los sistemas de herencia en el procesamiento del lenguaje natural puede encontrarse en [Daelemans et al., 1992].

El uso de las redes de herencia en el procesamiento del lenguaje natural proviene de varias tradiciones distintas. Por un lado, las "redes semánticas" utilizadas en Inteligencia Artificial [Sowa, 1984, Touretzky, 1986]. Por otro, el uso de jerarquías de tipos en modelos de computación [Ait-Kaci, 1984]. Por último, en la representación de los datos dentro del paradigma de programación orientada a objetos, en el que se basan lenguajes como Common Lisp Object System (CLOS) [Bobrow et al., 1990] y C++ [Stroustrup, 1992].

La idea, esencialmente, es encapsular la información común a varios objetos, de forma que no deba aparecer explícitamente en cada objeto, sino que la reciban implícitamente a través de algún mecanismo de herencia. Esta noción es particularmente útil en la especificación de información léxica. Todos los verbos, por ejemplo, comparten una serie de rasgos comunes que pueden ser abstraidos en una *clase* o en una *restricción de tipo* de la que cada verbo hereda esos rasgos. Los subtipos de un tipo dado especializan esa información de acuerdo con una distinción más fina. La clase de los verbos podría tener como subclases, por ejemplo, a los verbos transitivos por un lado y a los intransitivos por otro. Un verbo intransitivo heredaría la información correspondiente a la clase de los verbos intransitivos y a todas las superclases de ésta.

En las redes de *herencia simple*, cada tipo tiene a lo sumo un supertipo directo. En las redes de *herencia múltiple*, un tipo puede tener varios padres o supertipos directos. Por ejemplo, el verbo "caminar" podría heredar directamente de la clase de los verbos regulares y de la clase de los verbos de la primera conjugación.

El poder de estas generalizaciones es mucho mayor cuando las relaciones jerárquicas están reguladas mediante *herencia por defecto*, es decir, cuando la información correspondiente a un tipo puede ser anulada por otra información más específica, es decir, que aparezca más abajo en la jerarquía. Este tipo de herencia introduce, sin embargo, procesos no monótonos en los sistemas de representación, lo que complica el estudio teórico [Selman and Levesque, 1989]. En la sección 5 estudiamos estos problemas aplicados a nuestra propuesta, necesariamente no monótona.

Desde el punto de vista de la representación del conocimiento lingüístico, es obvio que la utilización de jerarquías y herencia es capaz de capturar generalizaciones lingüísticamente relevantes, como se prueba en teorías gramaticales como HPSG [Pollard and Sag, 1994]. Desde un punto de vista computacional, en [Daelemans et al., 1992] se citan algunas de las ventajas de los sistemas de representación léxica basados en herencia:

- Parsimonia. Un léxico descrito en torno a una red de herencia puede ser de un tamaño varios órdenes de magnitud menor que su equivalente sin estructura jerárquica.
- Facilidad de mantenimiento. Los cambios y adiciones se hacen en el nodo apropiado, no en cientos de entradas léxicas.
- Uniformidad en en la codificación de los distintos niveles de descripción lingüística.
- Modularidad.

La introducción de los procesos de herencia se suele hacer a través de una jerarquía de tipos en la que cada tipo lleva asociada una estructura de rasgos. Cada estructura de rasgos, recíprocamente, lleva asociado un tipo, de forma que las restricciones expresadas en las estructuras de rasgos correspondientes a ese tipo y a sus supertipos se transmiten a la entrada léxica mediante Unificación. Así, un tipo subsume a todas las estructuras de rasgos que heredan de él. Este tipo de sistemas se llaman sistemas de *estructuras de rasgos tipadas*. Herramientas de representación del conocimiento como ALE [Carpenter, 1992], TFS, CUF [Dörre and Dorna, 1993] o Troll [Gerdemann and King, 1993], pertenecen a esta clase de sistemas. Las jerarquías de estructuras de rasgos tipadas inducen restricciones sobre las estructuras permisibles Estas restricciones pueden utilizarse, además de como una forma de describir objetos, para realizar *inferencia de tipos*. El proceso de inferencia de tipos permite clasificar una estructura de rasgos dentro

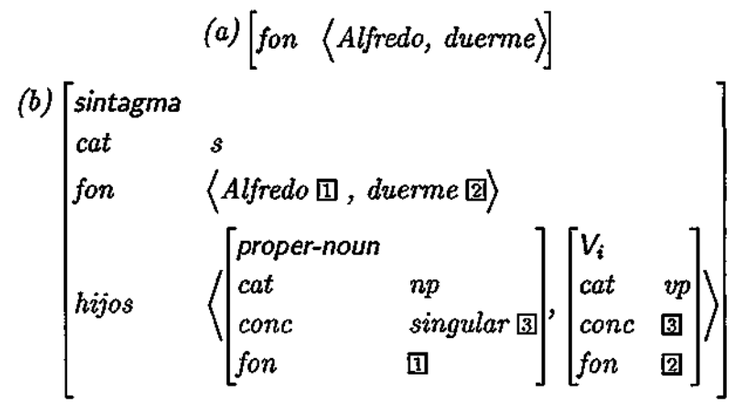


Figura 3.1: Mediante un proceso de inferencia de tipo, un sistema de estructuras de rasgos tipadas puede pasar de la estructura (a) a la estructura (b), buscando la estructura completa respecto a la jerarquía que cumple las restricciones de (a).

de la jerarquía de descripciones lingüísticas. Si la entrada de ese proceso es una estructura en la que sólo está especificada la cadena que le corresponde, el efecto del proceso de inferencia de tipos puede ser todo un proceso de parsing sobre esa cadena. En la figura 3.1 puede verse un ejemplo de este proceso.

Formalmente, una jerarquía de tipos viene definida por un conjunto finito de símbolos $\mathcal{T}$, que llamaremos tipos, y un orden parcial $\leq$ definido sobre ellos. Utilizaremos el término *poset* (partially ordered set), o conjunto parcialmente ordenado, para la tupla $\langle \leq, \mathcal{T} \rangle$. La relación de orden $\leq$ define la relación de subtipo: diremos que A es subtipo de B cuando A $\leq$ B. Llamaremos *tipos mínimos* o *terminales* a aquellos que no tienen subtipos. Suele considerarse dentro de toda jerarquía un tipo máximo $\top$, que representa información totalmente inespecífica, y un tipo mínimo $\bot$ que representa información contradictoria.

Una *estructura de rasgos tipada* (typed feature structure o tfs) se puede definir de forma similar a una estructura de rasgos como una tupla $F = \langle Q, q_0, \delta, \alpha \rangle$, donde la diferencia fundamental es que ahora α es una función total $\alpha : Q \rightarrow \mathcal{T}$ que da un tipo a cada nodo. El tipo de una tfs puede definirse como el tipo de su nodo raiz: $Tipo(F) = \alpha(q_0)$. En la figura 3.2 puede verse un ejemplo de estructura de rasgos tipada, tomado de [Copestake, 1992].

La definición de subsunción es equivalente a la que existía para estructuras de rasgos, con la condición adicional de que el tipo de la estructura subsumida ha de ser un subtipo de la estructura que subsume. La Unificación de dos tfss sigue siendo la máxima tfs subsumida por ambas en el ordenamiento establecido por la relación de subsunción. Por tanto, para que dos tfss puedan unificarse es necesario

$$\text{sheep} : \begin{bmatrix} \text{lex-noun-sign} & \\ \text{ORTH} & \text{sheep} \\ \text{COUNT} & \text{true} \\ \text{QUALIA} & \begin{bmatrix} \text{animal} & \\ \text{FORM} & \text{individual} \\ \text{SEX} & \text{gender} \end{bmatrix} \end{bmatrix}$$

Figura 3.2: Una estructura de rasgos tipada en LKB. El tipo de cada subestructura aparece en su parte superior.

que exista un máximo subtipo común a sus tipos respectivos. Supondremos que dados dos tipos t_1, t_2, existe a lo sumo un máximo subtipo común para los dos.

A cada tipo se asocia con una clase de estructuras de rasgos, mediante la definición de un conjunto de rasgos asociado a ese tipo que actúan como restricciones sobre las tfs que pertenecen a ese tipo. De esta forma se restringe el espacio de estructuras de rasgos admisibles, a diferencia de sistemas como PATR-II y FUG, donde cualquier estructura de rasgos era igualmente válida.

3.1.1 Reglas como estructuras de rasgos tipadas

En un sistema de estructuras de rasgos tipadas, las reglas gramaticales pueden introducirse como parte de las restricciones asociadas a la jerarquía de tipos. Mientras que en FUG las estructuras de rasgos que representaban las reglas estaban relacionadas mediante disyunciones, ahora es posible relacionarlas como subtipos de un tipo común.

Veamos un ejemplo en el que la única información que se hereda es la equivalente a una gramática libre de contexto. Consideremos de nuevo el conjunto de reglas:

$$\begin{aligned} &\text{VP} \rightarrow \text{V}_i \\ &\text{VP} \rightarrow \text{V}_t\ \text{NP} \\ &\text{VP} \rightarrow \text{VP PP} \end{aligned} \tag{3.1}$$

Así como en FUG estas tres reglas corresponderían a la disyunción de tres estructuras de rasgos, en un sistema de estructuras tipadas podrían expresarse

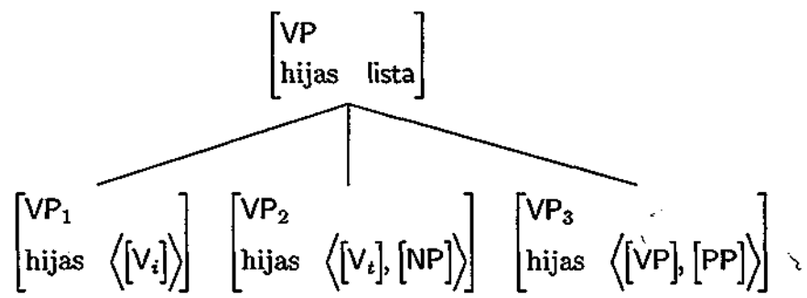


Figura 3.3: Tres reglas expandiendo un predicado verbal pueden representarse en una jerarquía de estructuras de rasgos tipadas como restricciones a tres subtipos directos del tipo VP.

mediante la jerarquía de tipos introduciendo tres subtipos VP_1, VP_2, VP_3 del tipo VP. En la estructura de rasgos que representa las restricciones asociadas a cada subtipo se podría incluir la información estructural equivalente a esas reglas como se muestra en la figura 3.3.

Usualmente, a las jerarquías de tipos se les da una interpretación de *universo cerrado*. La jerarquía prescribe un universo cerrado cuando cada elemento pertenece a un tipo mínimo. Dicho de otra forma, un objeto que pertenece a un tipo t ha de pertenecer necesariamente a alguno de los subtipos de t. De esta forma, la jerarquía de la figura 3.3 prescribe que un predicado verbal ha de estar formado según las tres reglas de la gramática libre de contexto equivalente en 3.1.

Esta disposición nos permite hablar de jerarquías de reglas. La estructura de rasgos correspondiente al tipo VP, por ejemplo, podría dar cuenta de determinadas restricciones que se aplicarán sobre la formación de cualquier VP. Los distintos subtipos de VP especializarían esta información. Este tipo de representación es muy útil para expresar principios gramaticales o restricciones generales sobre la formación de sintagmas. También es especialmente adecuada para representar sistemas en los que la mayor parte de la información recae sobre el léxico, mientras que un número reducido de reglas gobierna la transmisión de esta información. Su uso es especialmente adecuado en la representación computacional de HPSG, cuyo dominio informacional es precisamente el de las estructuras de rasgos tipadas. Por ejemplo, un principio gramatical de HPSG es que el rasgo de cabecera (*head feature*) de un sintagma ha de coincidir con el rasgo de cabecera de su *head-daughter* o componente de cabecera. Esta restricción puede representarse mediante un tipo head-feature-principle con una estructura de rasgos asociada que incluya la subestructura:

$$\begin{bmatrix} \textit{synsem} \mid \textit{loc} \mid \textit{cat} \mid \textit{head} & \boxed{1} \\ \textit{daughters} \mid \textit{head-daughters} \mid \textit{synsem} \mid \textit{loc} \mid \textit{cat} \mid \textit{head} & \boxed{1} \end{bmatrix}$$

El tipo head-feature-principle se introduce entonces como un supertipo de phrase, para así imponer ese principio sobre todos los objetos de tipo phrase [1]. En la figura 3.4 puede verse un fragmento de la jerarquía relacionada con la descripción de los sintagmas.

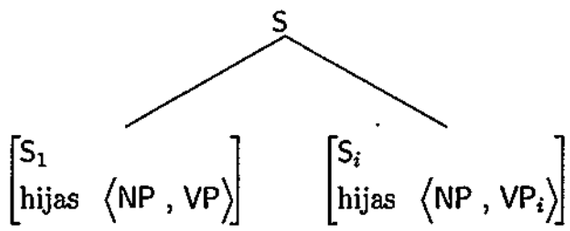
3.1.2 Inadecuación para representar reglas por defecto

Respecto a nuestro objetivo de representar el comportamiento excepcional, las jerarquías de estructuras de rasgos tipadas no nos permiten dar cuenta de los fenómenos de bloqueo a los que hemos hecho referencia en la introducción. Por ejemplo, en el caso de las dos reglas libres de contexto

$$\begin{aligned} S &\rightarrow NP\ VP \\ S_i &:= NP\ VP_i \end{aligned}$$

queríamos dar a $S_i := NPVP_i$ una interpretación excepcional respecto a $S \rightarrow$ NP VP, a saber, que un predicado nominal se combina con un predicado verbal para dar una oración excepto si el predicado verbal pertenece al subtipo VP_i, en cuyo caso se combina para dar un sintagma de tipo S_i.

Por un lado , la jerarquía equivalente a la interpretación usual de ese par de reglas sería ésta:



Sin embargo, el disponer de tipos asociados a las reglas nos permite jerarquizar nuestras dos reglas de otra forma, dando el primer paso hacia la especificación

[1] Este tipo de explicación es una simplicación de los mecanismos expresivos que requiere una teoría del tipo de HPSG. Ver [Pollard and Sag, 1994] para una descripción detallada de los mecanismos de representación de HPSG, y [Gotz and Meurers, 1995] para una discusión de las implicaciones computacionales de este tipo de representaciones en relación con las estructuras de rasgos tipadas y las relaciones entre ellas.

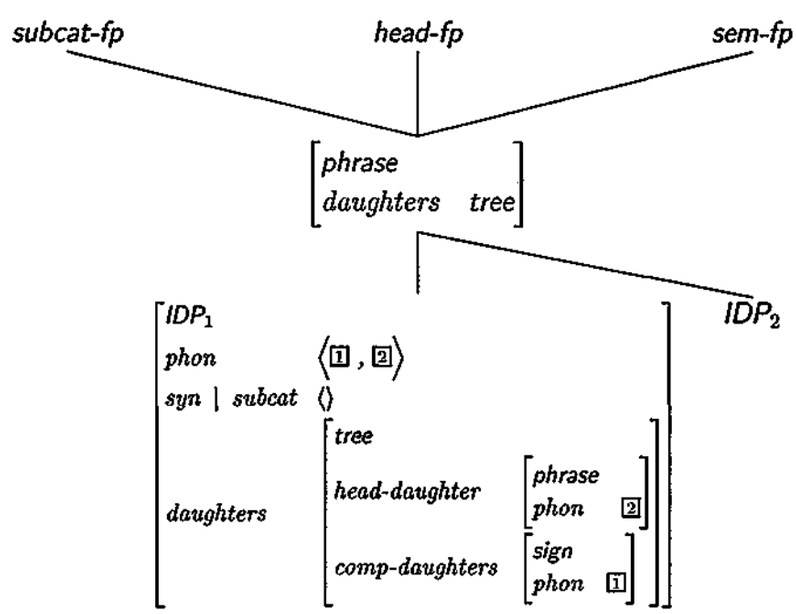


Figura 3.4: Parte de la jerarquía asociada al tipo phrase en HPSG. Cada tipo lleva asociado un conjunto de restricciones. En la figura hemos representado sólo las relacionadas con el tipo phrase y con su subtipo IPD_1. Todo objeto que pertenezca al tipo phrase, por ejemplo, ha tener un atributo *daughters* cuyo valor sea una estructura de rasgos de tipo tree. El tipo IDP_1 codifica la regla de HPSG que dice que un sintagma saturado (es decir, con la lista de subcategorización vacía) es la combinación de un sintagma insaturado con un complemento a su izquierda. Esta regla se aplica, por ejemplo, a oraciones formadas a partir de un VP (el sintagma insaturado) con un NP a su izquierda (el complemento).

del comportamiento excepcional de la una con respecto a la otra. Para ello las reorganizariamos así:

$$\begin{bmatrix} \mathsf{S} \\ \text{hijas} \quad \langle \mathsf{NP} \, , \, \mathsf{VP} \rangle \end{bmatrix}$$

$$\Big|$$

$$\begin{bmatrix} \mathsf{S}_i \\ \text{hijas} \quad \langle \mathsf{NP} \, , \, \mathsf{VP}_i \rangle \end{bmatrix}$$

Ahora la regla $\mathsf{S}_i := \mathsf{NP}\ \mathsf{VP}_i$, a la que queremos dar un sentido excepcional, guarda una relación de mayor especificidad con respecto a $\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP}$. La descripción no está mal formada, porque las especificaciones de tipos para las estructuras hijas del S_i es compatible con las de S. Sin embargo, la interpretación de esta jerarquía no es la deseada, ya que:

1. Si interpretamos la jerarquía como un universo cerrado, entonces todo objeto de tipo S debe pertenecer a alguno de los subtipos de S. Por lo tanto, todos los objetos de tipo S son, a su vez, de tipo S_i. Aunque hubiera otras reglas de producción para S, ningún objeto podría formarse directamente con la regla codificada en el tipo S. Por otro lado, la interpretación de universo cerrado es la que se adopta en todos los sistemas de estructuras de rasgos tipadas, y es fundamental en los procesos de inferencia de tipos y en los de disyunción.

2. Si interpretamos la jerarquía como un universo abierto, entonces puede haber objetos que pertenezcan directamente a S. Pero entonces las restricciones de S podrían aplicarse directamente sobre dos objetos de tipos NP, VP_i, sin necesidad de que el objeto resultante fuera de tipo S_i. No es pues, una solución.

3. Ni siquiera la forma en la que las reglas están jerarquizadas es la adecuada. En este caso, podría resultar razonable la jerarquización $\mathsf{S}_i \leq \mathsf{S}$. Pero la razón de que una regla sea más específica que otra no es la de que sus partes izquierdas tengan esa relación de especificidad, sino que está en relación con los objetos que combinan. Lo que queremos expresar es "Un predicado nominal se combina con un predicado verbal para dar una oración, *salvo que el predicado verbal sea del tipo particular* V_i, en cuyo caso se combinan en un objeto de tipo S_i. El tipo S_i podría no tener relación jerárquica alguna con S.

4. Por último, mientras las relaciones de herencia se interpreten mediante Unificación, no es posible establecer restricciones adicionales sobre la regla excepcional que anulen las restricciones sobre la regla por defecto. Este problema sería subsanable mediante algún tipo de Unificación por defecto si todos los establecidos anteriormente no impidieran la correcta interpretación de la jerarquía.

El aumentar la lógica de estructuras de rasgos mediante operadores adicionales como la disyunción, la negación o la implicación condicional ha sido propuesto en diversos trabajos [Pollard and Sag, 1987, Pollard and Sag, 1994, Carpenter and Pollard, 1991]. Sin embargo, la adición de esos operadores, como hemos comentado, complica considerablemente las propiedades computacionales de esos sistemas, y en general vuelve el procesamiento intratable (ver, por ejemplo, los trabajos recopilados en [Briscoe, 1991]). Y, en cualquier caso, no resuelven el problema de representación que estamos considerando.

3.2 Herencia por defecto versus reglas por defecto

El uso de la herencia por defecto ha sido una constante en la representación y mantenimiento de la información léxica desde hace más de una década [Fickinger, 1987, De Smedt, 1990, Daelemans, 1988, Daelemans and De Smedt, 1994, Copestake, 1992]. La herencia se usa no sólo en aras de una mayor economía descriptiva, sino también como una forma de capturar generalizaciones lingüísticamente relevantes. En la herencia monótona, un tipo sólo puede definirse sobre objetos que cumplen sin excepción todas sus restricciones. En la práctica, sin embargo, al definir una clase de palabras buscamos expresar las propiedades que se cumplen típicamente para los objetos que pertenecen a esa clase, dejando abierta la posibilidad de encontrar excepciones que contradigan esa información. El poder expresivo de las jerarquías es mucho mayor si se puede establecer información por defecto e información excepcional.

La mayoría de los sistemas de representación de información léxica no se basan en la Unificación, que es al fin y al cabo una operación estrictamente monótona. Es el caso de trabajos como los de [Fickinger, 1987] o el del sistema DATR [Evans and Gazdar, 1989a, Evans and Gazdar, 1989b], en el que se da cuenta de forma elegante de los procesos de blocaje asociados a las reglas de inflexión mediante una representación no monótona. Este tipo de formalismos son muy adecuados para la representación de información léxica, pero no nos sirven como mecanismos de representación de la información estructural asociada a las reglas.

En los últimos años, sin embargo, se han propuesto varias ope-

raciones no monótonas como variantes de la Unificación, que sustituirían a ésta en la interpretación de una jerarquía de estructuras de rasgos tipadas [Lascarides et al., 1994, Carpenter, 1993, Bouma, 1992, van den Berg and Prust, 1991, Calder, 1993]. La interacción entre la herencia múltiple y la unificación por defecto, así como la interacción de los valores reentrantes (o estructuras compartidas) con la unificación por defecto produce situaciones en las que no es fácil determinar cúal es la solución óptima. Este tipo de situaciones conflictivas es resuelto de forma muy distinta en las distintas aproximaciones que hemos citado.

Al contrario que en el caso de sistemas de representación léxica como DATR, en los formalismos léxicos basados en la Unificación por defecto existe la posibilidad de establecer jerarquías de reglas gramaticales entre las que existe una relación de Unificación por defecto. Es el caso de LKB [Copestake, 1992].

El mecanismo de reglas gramaticales de LKB (Lexical Knowledge Base) no supone extensión alguna al sistema de representación. Las reglas gramaticales y las reglas léxicas son estructuras de rasgos tipadas, que representan relaciones entre objetos. Las reglas léxicas se representan mediante el tipo rule, que tiene asociadas las restricciones:

$$\begin{bmatrix} rule & \\ 0 & sign \\ 1 & sign \end{bmatrix}$$

Todas las reglas léxicas son estructuras de rasgos de tipo rule o de algún subtipo de rule. Las reglas gramaticales son todas binarias y pertenecen al tipo grammar-rule:

$$\begin{bmatrix} grammar\text{-}rule & \\ 0 & sign \\ 1 & sign \\ 2 & sign \end{bmatrix}$$

Las reglas pueden contemplarse en LKB como una forma de generar nuevos objetos de tipo sign. Si dos signos F_1, F_2 pueden ser unificados con las estructuras en 1 y en 2, entonces la estructura de rasgos en 0 es un nuevo signo. En la LKB se asume que el orden en que aparecen los signos que se componen coincide con el orden numérico en que aparecen en la regla. Un parser puede operar sobre

este conjunto de reglas para analizar sintagmas u oraciones. En el proyecto ACQUILEX [Briscoe, 1991], que hace uso de LKB como sistema de representación básica, se codifica de esta forma una gramática del estilo de UCG (Unification Categorial Grammar, [Calder et al., 1988, Zeevat and Calder, 1987]) , que combina tres reglas gramaticales: *forward application, backward application* y *wrapping.*

Este tipo de aproximación ejemplifica el status de las reglas gramaticales dentro de los formalismos de representación lingüística más recientes. La información sobre constituencia se codifica en categorías recursivas al estilo de las gramáticas categoriales, y sólo existen tres reglas básicas de combinación sintáctica. A pesar de ser LKB un sistema de unificación, las reglas gramaticales no están embebidas como parte del sistema de restricciones, como se hace en FUG o TFS, sino que se describen como una clase de objetos distinta y son utilizadas mediante algoritmos de parsing conocidos. En LKB, esta decisión responde a los criterios que hemos comentado con anterioridad: las propiedades computacionales de las reglas gramaticales son distintas de las propiedades de los sistemas generales de restricciones, y la representación separada de estos dos tipos de información permite aplicar sobre cada uno los mecanismos computacionales más adecuados [2].

Las reglas gramaticales en LKB pueden heredar información por defecto a través de una jerarquía de reglas interpretada mediante el sistema de unificación por defecto descrito en [Lascarides et al., 1994]. Un buen ejemplo del uso exhaustivo de este recurso expresivo es el tratamiento semántico de los nombres compuestos en inglés en [Jones, 1992]. Jones propone una taxonomía de reglas según el predicado que se aplica entre los significados de los dos elementos del compuesto. Por ejemplo, una relación general es la de "tener", que tiene a su vez reglas específicas según el tipo de pertenencia: "posesión" (como en *student power*), "pertenencia" (*Jones house*), "composición" (*plastic cup*) ...

Por ejemplo, la regla correspondiente a "composición" se representa así en el lenguaje de LKB:

```
composition
grammar-rule
<>< have <>
< 0:sem:arg2:pred>= ''composed-of''
< 1: qualia > = physical
< 2: qualia > = physical.
```

La regla `composition` hereda por defecto las restricciones de `have`. Se aplica sobre dos objetos cuya cualidad es `physical`, y el predicado que relaciona sus semánticas es `composed-of`. La restricción `< 0:sem:arg2:pred>= ''composed-of''` sobreescribe la restricción

[2]En [Copestake, 1992] puede leerse: " It is unclear that the HPSG account of phrasal signs as feature structures which incorporate their daughters is the best one to adopt (..) There have to be good reasons to adopt an approach which makes most known parsing technology inapplicable"

```
< 0:sem:arg2:pred>= ''has''
```

asociada al tipo de regla have. De esta forma se puede establecer información por defecto asociada a las reglas.

Sin embargo, *no es lo mismo una regla por defecto que información por defecto asociada a una regla.* Para un sistema de parsing, cada regla gramatical de LKB tiene el mismo status, y puede aplicarse cuando se cumplen las restricciones sobre su parte derecha. No hay ningún mecanismo por el que la aplicación de una regla se bloquee cuando existe otra más especifica. Además, la especificidad de las reglas no está en relación directa con la especificidad de los argumentos. Esto produce un fenómeno de sobregeneración en la gramática de Jones. Para solucionar ese problema, Jones inserta en el proceso de parsing un mecanismo ad-hoc para eliminar los análisis espurios.

3.3 Recapitulación

Mientras que el concepto de información por defecto es utilizado profusamente en la representación de información léxica, no ha sido así en el caso de las reglas gramaticales. Sin embargo, el hecho de que la mayor parte de la información lingüística recaiga sobre el léxico hace que el número de reglas gramaticales sea muy pequeño y éstas muy generales, de forma que la posibilidad de establecer excepciones a estas reglas resulte especialmente conveniente.

Los formalismos de Unificación por defecto permiten jerarquizar las reglas de la gramática, y la transmisión por defecto de información entre ellas. La información estructural asociada a las reglas no tiene, sin embargo, una interpretación por defecto en ninguna de las variantes expresivas que hemos considerado (gramáticas libres de contexto aumentadas, disyunción de estructuras de rasgos, ramificación en una jerarquía de estructuras de rasgos tipadas). En ellos es posible expresar *información por defecto*, pero no *reglas por defecto* para combinar información.

Parte II

La propuesta: Reglas Genéricas

Capítulo 4

Reglas genéricas

Hemos visto hasta ahora cúal es el interés de disponer de un formalismo de representación y manejo de conocimiento lingüístico que permita expresar reglas por defecto, y porqué en los formalismos no-monótonos conocidos no se puede hacer.

Presentamos ahora un formalismo, el de las *reglas genéricas*, para dar una interpretación por defecto a las reglas gramaticales establecidas sobre una jerarquía lingüística de objetos. El comportamiento por defecto se obtendrá mediante un proceso de ligadura dinámica (inspirado en los lenguajes de programación orientados a objeto) situado en un punto intermedio entre el nivel de representación del conocimiento lingüístico y el nivel de los procesos de análisis.

Este formalismo permite desarrollar gramáticas de forma incremental, así como optimizar su tamaño y su mantenimiento. Veremos también cómo proporciona una interfaz para expresar la sintaxis y la semántica como procesos independientes pero intercomunicados, permitiendo una correspondencia más flexible entre reglas sintácticas y semánticas.

4.1 El modelo: polimorfismo en programación orientada a objetos

Los lenguajes de programación orientados a objeto se diferencian de los lenguajes convencionales tanto en la descripción de los datos como en la descripción de los programas que manejan esos datos.

Al igual que los sistemas de representación basados en estructuras de rasgos tipadas, la descripción de los datos en los lenguajes orientados a objeto se basa en el concepto de *herencia*. Según los lenguajes nos encontramos con herencia simple, herencia por defecto, herencia múltiple, etc. Por otro lado, la especificación de los programas explota esa representación particular de los datos mediante el concepto de funciones *polimórficas*. Pensamos que este concepto puede ser utilizado en la

descripción de reglas gramaticales, así como el concepto de herencia se utiliza en la descripción del léxico. La figura 4.1 ilustra el paralelismo que buscamos.

A continuación presentamos la concepción de los programas en torno a ese concepto de polimorfismo. No es mi intención hacer un repaso exhaustivo de las características de la programación orientada a objetos. Nos limitaremos a recoger las ideas que trasladaremos poseteriormente al terreno de la descripción lingüística.

En programación convencional, los argumentos de una función tienen un tipo definido. Si se invoca una función con argumentos de un tipo distinto al que se ha declarado, se produce un error o, al menos, su comportamiento es impredecible. En OOP la situación ha de ser necesariamente diferente, porque el conjunto de los tipos está estructurado mediante herencia, y cada tipo no tiene necesariamente una definición estanca. Efectivamente, en OOP es posible que una función admita datos de distintos tipos en cada uno de sus argumentos. A éste fenómeno se le llama *polimorfismo.*

Hay dos formas esenciales de tener una función polimórfica. La primera es consecuencia directa de la jerarquización de los tipos de datos: si se define una función sobre un tipo x, debería ser aplicable sobre todos los tipos de datos que sean subtipos de x. Este fenómeno es conocido como *genericidad.*

Por ejemplo, si se tiene definida una compleja jerarquía de tipos de ventanas, funciones como "activar ventana", no ncesitan ser definidas para cada subtipo de ventana, sino que basta con especificar una función con el tipo genérico ventana.

La otra forma de conseguir polimorfismo es mediante *sobrecarga.* Consiste en asociar varios procedimientos distintos a un mismo nombre de función, cada uno de ellos con argumentos de distinto tipo. Llamaremos *métodos* a este tipo de procedimientos, y *función genérica* (según la terminología de CLOS) a la función que forman. Por ejemplo, una función que calcule el área de una figura geométrica puede tener varios métodos distintos asociados a distintos tipos de figuras, como en el siguiente ejemplo en CLOS:

```
(defgeneric area ())

(defmethod area (x triangulo)
   (/ (* (base x) (altura x)) 2))

(defmethod area (x rectangulo)
    (* (base x) (altura x)))
```

Los métodos nunca se invocan directamente: se invoca la función genérica, y un mecanismo de *ligadura dinámica* decide qué método (o conjunto de métodos) ha de ser usado y de qué forma.

La combinación de genericidad y sobrecarga produce situaciones en las que

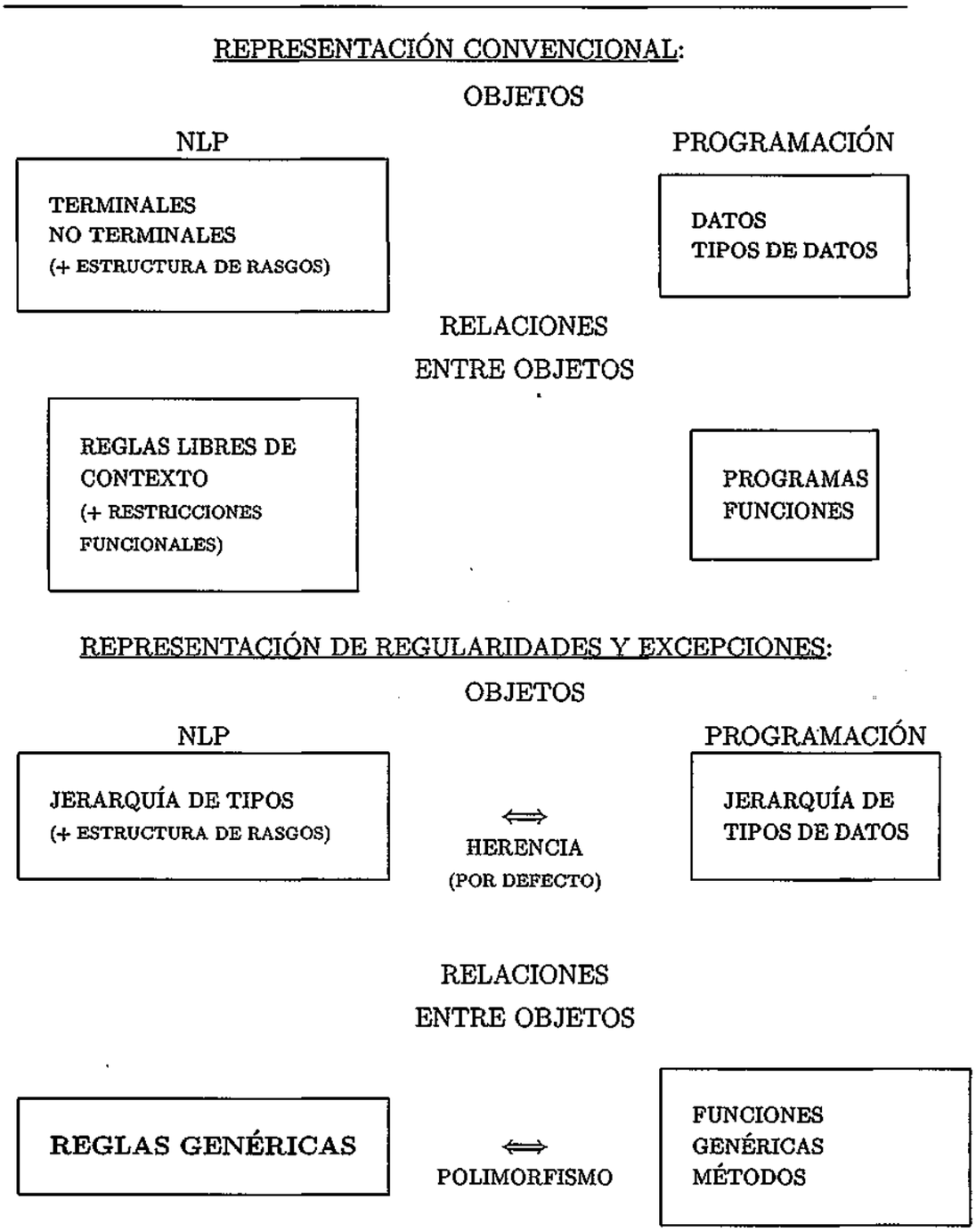


Figura 4.1: El formalismo de Reglas Genéricas completa el paralelismo entre los principios de la programación orientada a objetos y la representación del conocimiento lingüístico

existe más de un método que puede ser usado para unos argumentos concretos. Si se dota al sistema de un mecanismo capaz de establecer precedencia entre todos los métodos aplicables en una llamada concreta a la función y establecer la ligadura dinámica de acuerdo con esta precedencia, los distintos métodos de una función genérica pasan a tener una interpretación "por defecto". Es decir, un método se usará, en una llamada concreta, siempre que sea aplicable y no existan otros con mayor precedencia que él.

Es interesante reflexionar aquí sobre las diferencias que introduce este mecanismo de reglas por defecto frente a los que permiten las estructuras de rasgos tipadas (tfss). En éstas últimas, las reglas se pueden especificar implícitamente como compartición de estructuras entre la información correspondiente a cada uno de los hijos de un sintagma, como ya hemos visto en el capítulo anterior. Este tipo de especificaciones son polimórficas, en el sentido de que son válidas para todos los subtipos del tipo en el que se introducen. Sobre un sintagma dado pueden operar restricciones sintagmáticas expresadas a distintos niveles en la jerarquía léxica. Este polimorfismo, sin embargo, está limitado por el mecanismo computacional con el que se expresa: sigue siendo un fenómeno de herencia por defecto. Hay información más o menos específica, pero no mecanismos más o menos específicos de combinar información. El fenómeno de sobrecarga no existe como tal: las reglas son restricciones sobre tipos, y no están jerarquizadas entre ellas, sino a través de los tipos de cuya definición entran a formar parte. Esta limitación nos impide expresar reglas por defecto en el sentido que se estudia en esta memoria. Volveremos sobre este punto más adelante durante la descripción de nuestro formalismo.

En general, el criterio para seleccionar métodos al hacer la ligadura dinámica es el de escoger los métodos más específicos. El proceso tiene tres pasos. En primer lugar, selección de los métodos aplicables (es decir, aquellos cuyos argumentos tienen tipos compatibles con los de los argumentos de la llamada). En segundo lugar, asignación de prioridades. A continuación se aplica un algoritmo de combinación que nos proporciona el código efectivo que será aplicado sobre los argumentos. Normalmente ese algoritmo se reduce a seleccionar el método de máxima prioridad.

En el caso de funciones unarias, como en el ejemplo, la especificidad de un método equivale a la especificidad del tipo asociado. El caso de funciones de aridad superior a uno es diferente, y tiene una solución distinta en cada lenguaje de programación orientado a objetos. En CLOS, por ejemplo, la precedencia entre métodos se determina de acuerdo con el primer argumento de la función. Si hay dos métodos con tipos idénticos en el primer argumento, la precedencia entre ellos se establece de acuerdo con el segundo argumento. Si de nuevo los tipos son idénticos, se recurre al tercer argumento, y así sucesivamente hasta que se rompe la indeterminación. Este sistema es, por tanto, asimétrico respecto a los argumentos de la función.

En C++, sin embargo, el método escogido ha de ser más especifico que los

demás en cada uno de sus argumentos. Si para una tupla de argumentos dada no existe un método que cumpla esa condición, la función devuelve un error al ser invocada.

Una característica fundamental de ambos sistemas es su no monotonicidad: el proceso de ligamiento dinámico involucra casi siempre el descartar información genérica en favor de información más específica.

Los conceptos de polimorfismo y ligadura dinámica nos sugieren la propuesta para especificar reglas por defecto.

4.2 Propuesta

Partiremos de un supuesto: los objetos lingüísticos están modelados de acuerdo con una jerarquía de tipos. Hemos visto ya que este supuesto es razonable.

Proponemos para la representación de las operaciones asociadas a los objetos lingüísticos una aproximación *funcional* y *orientada a objetos*, en la que cada operación individual tiene una interpretación por defecto de acuerdo con un sistema adecuado de ligadura dinámica que establece la precedencia de acuerdo con la especificidad de los argumentos.

Llamaremos *regla parcial* a cada uno de los métodos que intervienen para combinar un tipo de información, y *regla genérica* al conjunto de esas reglas junto con las condiciones para su interpretación.

Posponiendo la definición formal para la sección siguiente, hagamos aquí una aproximación gradual a las reglas genéricas, estudiando la funcionalidad que deben tener.

Recordemos el problema que planteábamos en la introducción a esta memoria. Disponíamos de la siguiente jerarquía (expresada en la notación standard de gramáticas libres de contexto):

$$\mathsf{VP} \rightarrow \mathsf{VP}_1 \mid \mathsf{VP}_2 \mid \mathsf{VP}_3 \mid \mathsf{VP}_4 \mid \mathsf{VP}_i \tag{4.1}$$

y también de la siguiente regla:

$$\mathsf{S} \rightarrow \mathsf{NP}\ \mathsf{VP} \tag{4.2}$$

Encontrábamos que la combinación de objetos VP_i debía hacerse de una forma específica, y queríamos reflejar este hecho mediante un nuevo subtipo $\mathsf{S} \rightarrow \mathsf{S}_i$ y una nueva regla:

$$\mathsf{S}_i \rightarrow \mathsf{NPVP}_i \tag{4.3}$$

Sin embargo, esta regla no era suficiente para expresar que la combinación de un NP con un objeto de tipo específico VP_i debe hacerse a través de la regla 4.3 y no a través de la regla 4.2.

Vamos, pues, a reescribir estas reglas de manera funcional, interrogándonos sobre las condiciones en que tienen el comportamiento deseado.

La regla enunciada en 4.2, junto con la jerarquía 4.1 tiene esta forma como regla genérica:

$$\mathcal{SYN} = \{\ SYN_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{x},\mathsf{y}) = \mathsf{S} \tag{4.4}$$

Mediante esta notación hemos especificado una regla genérica $\mathcal{SYN}$ que combina dos categorías en una tercera, si esto es posible. De momento, esta regla genérica se compone de una única *regla parcial*, $SYN_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{x},\mathsf{y}) = \mathsf{S}$. La notación $SYN_{\mathsf{NP}\otimes\mathsf{VP}}$ significa que esa regla es aplicable cuando el primer argumento es de tipo NP y el segundo de tipo VP, es decir, cuando $\mathsf{X} \leq \mathsf{NP}$ y $\mathsf{Y} \leq \mathsf{VP}$. La genericidad viene del hecho de que los argumentos no han de ser exactamente de un tipo dado, sino de cualquiera de los subtipos de un tipo dado.

Para añadir la regla excepcional 4.3 , simplemente ampliamos $\mathcal{SYN}$:

$$\mathcal{SYN} = \begin{cases} SYN_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = \mathsf{S} \\ SYN_{\mathsf{NP}\otimes\mathsf{VP}_i}(\mathsf{X},\mathsf{Y}) = \mathsf{S}_i \end{cases} \tag{4.5}$$

Cuando, en un proceso de parsing , se desee combinar sintácticamente dos categorías, se invocará la función genérica $\mathcal{SYN}$. Entonces necesitaremos un proceso de ligadura dinámica que 1) determine las reglas parciales aplicables y 2)seleccione la combinación de reglas más adecuada atendiendo a su especificidad. En nuestro caso, este proceso de selección se reduce a tomar la más específica.

¿Cúal será el criterio adecuado por el que establecer la precedencia entre reglas parciales? El más evidente es considerar que una regla parcial $R_{\mathsf{a}_1\otimes\mathsf{a}_2}$ es más específica que otra $R_{\mathsf{b}_1\otimes\mathsf{b}_2}$ si, y sólo si, es más específica en cada uno de sus argumentos: $a_1 \leq b_1, a_2 \leq b_2$.

Veamos algunos ejemplos del funcionamiento de $\mathcal{SYN}$:

$$\begin{array}{l} \mathcal{SYN}(\mathsf{NP},\mathsf{VP}) = \mathsf{S} \\ \mathcal{SYN}(\mathsf{NP},\mathsf{VP}_2) = \mathsf{S} \\ \mathcal{SYN}(\mathsf{VP}_2,\mathsf{NP}) = \bot \\ \mathcal{SYN}(\mathsf{NP},\mathsf{VP}_i) = \mathsf{S}_i \end{array} \tag{4.6}$$

Las reglas parciales que se han usado en cada caso son:

$$\begin{array}{l} \mathcal{SYN}(\mathsf{NP},\mathsf{VP}) \Longrightarrow R_{\mathsf{NP}\otimes\mathsf{VP}} \\ \mathcal{SYN}(\mathsf{NP},\mathsf{VP}_2) \Longleftrightarrow R_{\mathsf{NP}\otimes\mathsf{VP}} \\ \mathcal{SYN}(\mathsf{VP}_2,\mathsf{NP}) \Longleftrightarrow \text{No hay ninguna regla aplicable} \\ \mathcal{SYN}(\mathsf{NP},\mathsf{VP}_i) \Longleftrightarrow R_{\mathsf{NP}\otimes\mathsf{VP}} \end{array} \tag{4.7}$$

De esta forma conseguimos especificar reglas por defecto y reglas con distintos grados de especificidad sin necesidad de diferenciarlas y sin necesidad de establecer de antemano las dependencias entre ellas. Conseguimos así la mayor flexibilidad para desarrollar gramáticas incrementalmente.

Un punto muy interesante de las reglas genéricas es la interfaz sintaxis-semántica que puede desarrollarse mediante ellas. Si se representan las operaciones de combinación semántica mediante una regla genérica, la relación entre reglas sintácticas y semánticas se vuelve totalmente flexible: no hay necesidad de establecer una biyección en la gramática entre reglas sintácticas y semánticas. Varias situaciones en las que se aplica una misma regla semántica pueden corresponderse con varias reglas sintácticas y viceversa. Esta es una posibilidad que se ha defendido en aproximaciones recientes como la de [Bos et al., 1994].

Para ilustrar este tipo de interacción, consideremos la siguiente regla genérica de combinación semántica:

$$\mathcal{SEM} = \begin{cases} \mathit{SEM}_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = \mathsf{X}(\mathsf{Y}) \\ \mathit{SEM}_{\mathsf{nombre\text{-}propio}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = \mathsf{Y}(\mathsf{X}) \wedge (\lambda P.P(\mathsf{Alberto}))(\mathsf{Y}) \end{cases} \tag{4.8}$$

Suponemos que cada objeto tiene asociada una interpretación semántica en forma de expresión de un cálculo λ. Esta regla genérica especifica que la combinación semántica de predicados nominales con predicados verbales se hace mediante aplicación funcional de la semántica del predicado verbal sobre la semántica del predicado nominal. La excepción es la regla parcial $\mathit{SEM}_{\mathsf{nombre\text{-}propio}\otimes\mathsf{VP}}$, que refleja una propiedad excepcional del entorno semántico, a saber: que alguien llamado Alberto imita las actividades de todos los que le rodean en ese universo. Hemos introducido, para establecer esta regla, un subtipo nombre-propio de NP.

Consideremos los siguientes objetos lingüísticos:

- Nombre-Propio Ana : $\lambda P.P(\text{Ana})$
- NP El+perro : $\lambda P.\forall x[\text{PERRO}(x) \rightarrow P(x)]$
- VP_i se+enfadó : $\lambda x.\text{ENFADADO}(x)$
- VP_1 come : $\lambda x.COME(x)$

Mediante las reglas genéricas $\mathcal{SYN}$ y $\mathcal{SEM}$ pueden obtenerse las siguientes descripciones:

- S El+perro+come : $\forall x[\text{PERRO}(x) \rightarrow COME(x)]$
- S Ana+come : $\text{COME(Ana)} \wedge \text{COME(Alberto)}$
- S_i Ana+se+enfadó : $\text{ENFADADO(Ana)} \wedge \text{ENFADADO(Alberto)}$
- S_i El+perro+se+enfadó : $\forall x[\text{PERRO}(x) \rightarrow \text{ENFADADO}(x)]$

Las reglas parciales que han intervenido en la formación de estos sintagmas son:

- El+perro+come : $\begin{cases} SEM_{\text{NP}\otimes\text{VP}} \\ SYN_{\text{NP}\otimes\text{VP}} \end{cases}$
- Ana+come : $\begin{cases} SEM_{\text{Nombre-Propio}\otimes\text{VP}} \\ SYN_{\text{NP}\otimes\text{VP}} \end{cases}$
- Ana+se+enfadó $\begin{cases} SEM_{\text{Nombre-Propio}\otimes\text{VP}} \\ SYN_{\text{NP}\otimes\text{VP}_i} \end{cases}$
- El+perro+se+enfadó $\begin{cases} SEM_{\text{NP}\otimes\text{VP}} \\ SYN_{\text{NP}\otimes\text{VP}_i} \end{cases}$

En la figura 4.2 puede verse la comparación entre la combinación de las dos reglas genéricas $\mathcal{SYN}$ y $\mathcal{SEM}$ y una gramática sintagmática equvalente. Si se reescribe el conjunto de categorías para minimizar el número de reglas sintagmáticas equivalentes a dos reglas genéricas actuando entrelazadas, se necesitan hasta $n*m$ reglas para conseguir una gramática equivalente a dos reglas genéricas con n y m reglas respectivamente.

Gramática por defecto:

$$\mathcal{SYN} = \begin{cases} SYN_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = \mathsf{S} \\ SYN_{\mathsf{NP}\otimes\mathsf{VP}_i}(\mathsf{X},\mathsf{Y}) = \mathsf{S}_i \end{cases}$$

$$\mathcal{SEM} = \begin{cases} SEM_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = sem(\mathsf{X})(sem(\mathsf{Y})) \\ SEM_{\mathsf{Nombre\text{-}Propio}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = sem(\mathsf{X})(sem(\mathsf{Y})) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{Y})) \end{cases}$$

Gramática sintagmática equivalente:

R1: $\mathsf{S}_i \rightarrow \mathsf{NP}^*\mathsf{VP}_i$
$\mathtt{sem}(\mathsf{S}_i) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_i))$
R2: $\mathsf{S}_i \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_i$
$\mathtt{sem}(\mathsf{S}_i) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_i)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_i))$
R3: $\mathsf{S} \rightarrow \mathsf{NP}^*\mathsf{VP}_1$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_1))$
R4: $\mathsf{S} \rightarrow \mathsf{NP}^*\mathsf{VP}_2$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_2))$
R5: $\mathsf{S} \rightarrow \mathsf{NP}^*\mathsf{VP}_3$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_3))$
R6: $\mathsf{S} \rightarrow \mathsf{NP}^*\mathsf{VP}_4$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_4))$
R7: $\mathsf{S} \rightarrow \mathsf{NP}^*\mathsf{VP}_5$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{NP}^*)(sem(\mathsf{VP}_5))$
R8: $\mathsf{S} \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_1$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_1)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_1)))$
R9: $\mathsf{S} \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_2$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_2)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_2)))$
R10: $\mathsf{S} \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_3$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_3)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_3)))$
R11: $\mathsf{S} \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_4$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_4)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_4)))$
R12: $\mathsf{S} \rightarrow \mathsf{Nombre\text{-}Propio}\mathsf{VP}_5$
$\mathtt{sem}(\mathsf{S}) = sem(\mathsf{Nombre\text{-}Propio})(sem(\mathsf{VP}_5)) \wedge (\lambda P.P(\mathsf{Alberto})(sem(\mathsf{VP}_5)))$

Figura 4.2: *Comparación entre una gramática de reglas genéricas y su equivalente sintagmático.* En la segunda, es necesario introducir un nuevo tipo $\mathsf{NP}^* \leq \mathsf{NP}$ para los objetos que pertenecen al tipo NP pero no a Nombre-Propio.

4.3 Formalización

Ha llegado el momento de dar una definición formal de regla genérica, que dote de un contenido preciso a la simbología y los conceptos que hemos esbozado en la sección anterior.

La definición debe girar en torno a los siguientes elementos:

1. Un dominio jerarquizado de objetos lingüísticos.
2. Un dominio informacional.
3. Una forma de asignar precedencia a las reglas.
4. Un mecanismo de ligadura dinámica.

A continuación ofrecemos esa definición, introducida en [Gonzalo, 1995].

Definición de regla genérica

Consideremos un conjunto parcialmente ordenado (poset) $\langle \leq, \mathcal{T} \rangle$ y un dominio de objetos lingüísticos $\mathcal{O}$ tipado con $\langle \leq, \mathcal{T} \rangle$ (es decir, existe una función $\texttt{tipo}{:}\mathcal{O} \rightarrow \mathcal{T}$). Consideremos además un dominio informacional Φ asociado a $\mathcal{O}$ mediante la función $\phi{:}\mathcal{O} \rightarrow \Phi$, que asocia al menos un elemento de Φ a cada objeto de $\mathcal{O}$ y que puede, en principio, ser multivaluada.

Una **regla genérica** $\mathcal{G}$ *sobre la tupla* $\langle \langle \leq, \mathcal{T} \rangle, \mathcal{O}, \Phi \rangle$ *es otra tupla* $\langle \langle \preceq, \mathcal{T}^* \rangle, \mathcal{F}, \mathrm{G} \rangle$ *donde*

- $\mathcal{F}$ *es un conjunto de funciones, que llamaremos* **reglas parciales**, *de la forma*

$$f_{t_1 \otimes t_2} : \Phi \times \Phi \rightarrow \Phi$$

  donde $t_1, t_2 \in \mathcal{T}$, *y ha de cumplirse* $\forall f_{t_1 \otimes t_2}, f_{q_1 \otimes q_2} \in \mathcal{F}, t_1 = q_1 \wedge t_2 = q_2 \Longleftrightarrow f_{t_1 \otimes t_2} = f_{q_1 \otimes q_2}$.

- $\langle \preceq, \mathcal{T}^* \rangle$ *es un conjunto parcialmente ordenado definido sobre* $\mathcal{F}$ *de la siguiente manera:*

$$\mathcal{T}^* \equiv \{t_1 \otimes t_2 | f_{t_1 \otimes t_2} \in \mathcal{F}\}$$
$$\forall x_1 \otimes x_2, y_1 \otimes y_2 \in \mathcal{T}^* \quad x_1 \otimes x_2 \preceq y_1 \otimes y_2 \Longleftrightarrow x_1 \leq y_1 \text{ and } x_2 \leq y_2$$

Llamaremos **poset cartesiano** $\langle \preceq, \mathcal{T}^* \rangle$ *sobre un poset* $\langle \leq, \mathcal{T} \rangle$ *a cualquier subconjunto de* $\mathcal{T} \times \mathcal{T}$ *dotado con la relación de orden parcial* $\preceq$.

Llamaremos **tipo cartesiano** *de una regla parcial* $f_{t_1 \otimes t_2}$ *a* $t_1 \otimes t_2$.

- $\forall x_1, x_2 \in \mathcal{T}, \mathbf{G}(x_1, x_2) \equiv f_{t_1 \otimes t_2}$ *tal que* $t_1 \otimes t_2$ *es la mínima cota superior de* $x_1 \otimes x_2$ *en* $\mathcal{T}^* \cup \{x_1 \otimes x_2\}$.

  G *selecciona la regla parcial más específica de* $\mathcal{F}$ *dados dos argumentos de tipos* x_1, x_2.

Nótese que $\mathcal{T}^*$ es un subconjunto del producto cartesiano $\mathcal{T} \times \mathcal{T}$, y que la relación de orden parcial $\preceq$ se deduce de la relación $\leq$ que existe sobre los elementos de $\mathcal{T}$. Mediante la definición de *poset cartesiano* hemos conseguido axiomatizar la precedencia entre reglas parciales, convirtiendo el mecanismo de ligadura dinámica en una verificación de dependencias en un orden parcial. Compárese con los mecanismos de ligadura dinámica en CLOS y C++, nuestros ejemplos paradigmáticos de programación orientada a objetos: en ambos, la ligadura dinámica se hace a través de especificaciones algorítmicas, y todo el ordenamiento de los métodos se hace dinámicamente, es decir, no existe un preorden como el que nosotros inducimos mediante un poset cartesiano.

La demostración de que la relación $\preceq$ define, efectivamente, un conjunto parcialmente ordenado, es trivial:

- $\preceq$ es reflexiva: $a_1 \otimes a_2 \preceq a_1 \otimes a_2$, ya que $a_1 \leq a_1$ y $a_2 \leq a_2$.
- $\preceq$ es antisimétrica: Si $a_1 \otimes a_2 \leq b_1 \otimes b_2$ y $b_1 \otimes b_2 \leq a_1 \otimes a_2$, entonces $a_1 \leq b_1, b_1 \leq a_1 \rightarrow a_1 = b_1$ y $a_2 \leq b_2, b_2 \leq a_2 \rightarrow a_2 = b_2$.
- $\preceq$ es transitiva: Si $a_1 \otimes a_2 \preceq b_1 \otimes b_2$ y $b_1 \otimes b_2 \preceq c_1 \otimes c_2$, entonces $a_1 \leq b_1 \leq c_1$ y $a_2 \leq b_2 \leq c_2$ y por tanto $a_1 \otimes a_2 \preceq c_1 \otimes c_2$.

Los posets aproximan las reglas parciales a las reglas expresables en sistemas de estructuras de rasgos tipadas, donde la especificidad de las reglas está implicita en la jerarquía que forman sus tipos. Hay, sin embargo, dos diferencias fundamentales:

1. La relación de orden parcial no se hace explícita al escribir las reglas, sino que se deduce de las condiciones sobre los argumentos de cada regla. El orden se alcanza de forma implícita, frente a los sistemas de estructuras de rasgos, en los que el orden se define al escribir cada regla individual.

2. En los sistemas de estructuras de rasgos, el orden entre reglas no es más que el orden entre los objetos lingüísticos que describen, es decir, entre la parte izquierda de esas reglas. Las reglas parciales, por el contrario, se ordenan de acuerdo con la especificidad de la información que están combinando, independientemente del tipo que tenga el objeto resultante.

La función G proporciona la interfaz de la regla genérica, como sistema de funciones orientadas a objeto, con el sistema o la gramática que lo invoca. En concreto, es a través de esta función, que llamaremos *función de ligadura dinámica*, como las reglas parciales reciben su interpretación de reglas por defecto. Dados un par de argumentos con tipos x_1, x_2, se considera dinámicamente una extensión $\mathcal{T}^* \cup \{x_1 \otimes x_2\}$ del poset cartesiano original $\langle \preceq, \mathcal{T}^* \rangle$ que incluye al nuevo tipo $x_1 \otimes x_2$. El cual, en virtud de la definición de la relación de orden $\preceq$, toma su lugar en la jerarquía. Sus supertipos denotan entonces todas las reglas que pueden ser aplicadas para componer dos argumentos de tipos x_1, x_2. Y el orden parcial entre esos supertipos denota la precedencia entre las reglas asociadas. La acción de la regla genérica G sobre un par de objetos lingüísticos de tipos x_1, x_2 y expresiones asociadas ϕ_1, ϕ_2 viene dada por $G(x_1, x_2)(\phi_1, \phi_2)$.

Hemos postulado aquí la definición más simple para G, que aplica simplemente la regla parcial más específica, dejando de lado el resto. Sin embargo, la definición es suficientemente modular a este respecto como para permitir variaciones en las que otros mecanismos fueran postulados para combinar las reglas aplicables teniendo en cuenta su precedencia. Podría demostrarse útil, por ejemplo, que si las reglas se expresaran como estructuras de rasgos, G aplicara algún tipo de unificación por defecto de todas las reglas aplicables. En este caso, habría de ser un mecanismo de unificación asimétrico, como puede ser el de [Bouma, 1992], para que el orden de precedencia entre las reglas tuviera efecto. No serían aplicables aproximaciones independientes del orden, como la de [Lascarides et al., 1994], donde la especificidad de la información que lleva asociada cada tipo es la que decide qué información prevalece, y no una decisión a priori sobre cual de los dos argumentos que intervienen en la unificación será interpretado como "información por defecto".

Nótese que no hemos precisado cómo es el dominio de objetos lingüísticos $\mathcal{O}$, y mucho menos cúal es el formalismo lingüístico en particular en el que está representado. El único requerimiento sobre $\mathcal{O}$ es que esté tipado. El único requerimiento sobre las operaciones que relacionan objetos, por otro lado, es que operen sobre sintagmas y sean binarias. Hemos adoptado la restricción de que las reglas sean binarias para poder aprovechar todas las capacidades del concepto de ligadura dinámica con las mínimas complicaciones. Sin embargo, ésta es una restricción poco relevante, por dos razones: (1) En los formalismos actuales la mayoría de las reglas son binarias, desde las gramáticas de unificación hasta casos extremos como el de las gramáticas categoriales, en los que sólo hay reglas binarias. (2) Cualquier gramática libre de contexto es reescribible en otra compuesta exclusivamente por reglas binarias (es decir, en forma normal de Chomsky).

$$
\begin{aligned}
&X_0 \rightarrow X_1\, X_2 \\
&\quad \langle X_0\, cat \rangle = s \\
&\quad \langle X_1\, cat \rangle = np \\
&\quad \langle X_2\, cat \rangle = vp \\
&\quad \langle X_0\, head \rangle = \langle X_2\, head \rangle \\
&\quad \langle X_0\, head\, subject \rangle = \langle X_1\, head \rangle \\
&\quad \langle X_0\, sem \rangle \leftarrow \mathcal{SEM}(X_1, X_2)
\end{aligned}
$$

Figura 4.3: Una posible forma de llamar a una regla genérica que nos devolviera la combinación semántica de los objetos involucrados en la regla de PATR-II. Hemos utilizado el símbolo $\leftarrow$ para indicar que no se trata de una restricción normal que haya de ser manejada mediante unificación, sino de la invocación de una regla genérica.

Las reglas genéricas pueden ser invocadas por gramáticas de cualquier tipo, siempre que el dominio linguístico jerarquizado, para relacionar dominios lingüísticos que así lo requieran. Por ejemplo, una gramática de unificación al estilo de PATR-II [Shieber, 1986] podría incluir, además de condiciones de unificación, llamadas a una regla genérica que hiciera la interpretación semántica, como en la figura 4.3 Este tipo de interacción tiene la contrapartida de que restringe la bidireccionalidad de las reglas de unificación, ya que las reglas genéricas, entendidas como funciones, son unidireccionales.

Capítulo 5

Precedencia y No monotonicidad

5.1 Conflictos de precedencia entre reglas parciales

En la sección anterior hemos obviado las cuestiones y problemas que surgen relacionados con el ordenamiento y la precedencia entre reglas parciales. Éstas merecen, en efecto, una sección aparte.

Veamos un ejemplo del tipo de conflictos que pueden darse. Consideremos la jerarquía $\langle \leq, \mathcal{T} \rangle$ de la figura 5.1. Consideremos, sobre ella, una regla genérica $\mathcal{G}$ cuyo poset cartesiano asociado sea $\langle \preceq, \mathcal{T}^* \rangle$ (también en la figura 5.1). En él se especifica una regla parcial aplicable sobre tuplas posesivo, nombre, y otra aplicable sobre tuplas pronombre, nombre-contable. Ésta es un situación potencialmente conflictiva, como puede verse si intentamos aplicar $\mathcal{G}$ para componer la información de la tupla sus,gallinas. Efectivamente, al crear $\langle \preceq, \mathcal{T}^* \cup \{\text{sus}\otimes\text{gallinas}\} \rangle$ para efectuar el proceso de ligadura dinámica, nos encontramos con que el tipo sus⊗gallinas presenta herencia múltiple con respecto a los tipos posesivo⊗nombre, pronombre⊗nombre-contable. No disponemos de ningún criterio para establecer precedencia entre estas dos reglas.

El mecanismo de ordenación que hemos incorporado a la definición de regla genérica, es decir, los posets cartesianos, es muy restrictivo. Recordemos que la regla más específica, para una tupla dada de argumentos con tipos x_1, x_2, es aquélla regla $f_{t_1 \otimes t_2}$ tal que $t_1 \otimes t_2$ es la mínima cota superior de $x_1 \otimes x_2$ en el conjunto $\mathcal{T}^* \cup \{x_1 \otimes x_2\}$. Sin embargo, esta cota puede no existir o no ser única. El caso de no existencia refleja simplemente que no hay ningún mecanismo en la gramática que permita esa operación de combinación. El caso de no unicidad, sin embargo, es más complejo, ya que deja indeterminado cúal es el procedimiento que debe usarse, si deben usarse los dos (que no serán, en general, compatibles), etc.

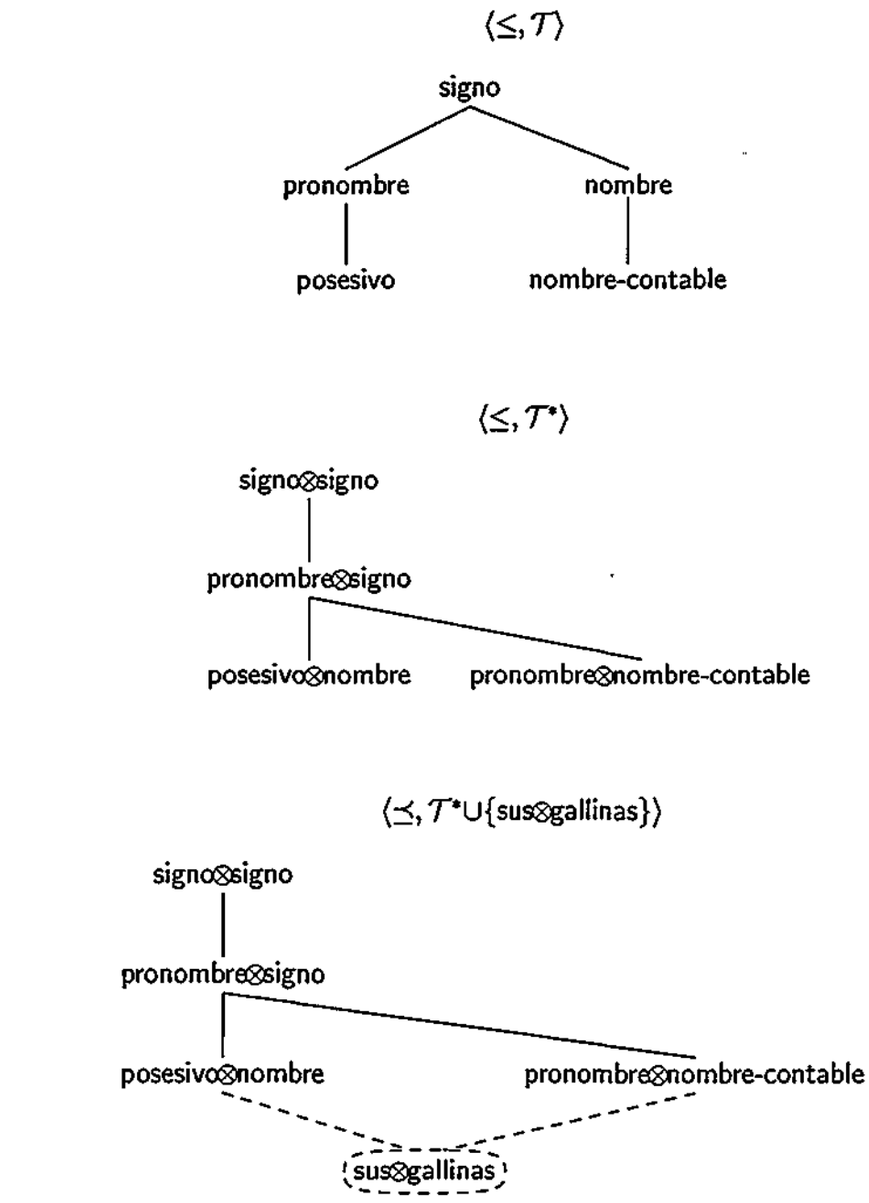


Figura 5.1: Si aplicamos la regla genérica descrita por el poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle$ a "sus gallinas", se produce una indeterminación: no existe una única regla más específica que el resto; el tipo sus⊗gallinas presenta herencia múltiple.

5.2 Origen de los problemas de precedencia

El problema presentado tiene su origen en la no monotonicidad del proceso de ligadura dinámica. Este tipo de problemas aparece hasta en los sistemas no monótonos más sencillos. Recordemos que una lógica no monótona es aquella en la que la inclusión de nuevos axiomas puede invalidar teoremas que eran demostrables a partir del conjunto primitivo de axiomas. Por ejemplo, en un sistema que incluyera los axiomas

- Las aves vuelan
- Los pingüinos son aves

podría demostrarse el teorema "Los pingüinos vuelan". Si añadimos al sistema el enunciado "Los pingüinos no vuelan", el teorema "Los pingüinos vuelan" dejará de ser válido. Si el sistema fuera monótono, la inclusión del enunciado "Los pingüinos no vuelan" sería simplemente una incoherencia.

Está claro que el formalismo de las reglas genéricas no es un sistema monótono. Al añadir una nueva regla parcial a una regla genérica, es probable que haya derivaciones no válidas que sí lo fueran en el sistema anterior. De hecho, esa es precisamente la interpretación que deseamos para una regla parcial: que sustituya a otras menos específicas, cambiando por tanto el sentido de los análisis donde es aplicable. Recordemos que la regla libre de contexto 4.2 permitía la derivación:

$$S \rightarrow NP\ VP \rightarrow NP\ VP_i \tag{5.1}$$

y, al añadir la regla 4.3, ésta derivación seguía siendo válida. Una gramática libre de contexto es, de hecho, un sistema monótono. Sin embargo, las mismas reglas expresadas como reglas parciales tenían un comportamiento diferente: mientras que la regla genérica 4.4 permitía la derivación anterior, la inclusión de una segunda regla parcial en 4.5 habilitaba la derivación

$$S \rightarrow S_i \rightarrow NP\ VP_i \tag{5.2}$$

pero impedía que se produjera la derivación anterior.

Este tipo de problemas aparece inevitablemente al introducir en un sistema de representación mecanismos de herencia no monótonos. Las soluciones que se adoptan para evitar los conflictos entre información contradictoria con el mismo nivel de especificidad son de dos clases: las restrictivas, que reducen la expresividad del sistema para evitar ese tipo de conflictos, y las que exigen introducir información adicional para tratar esos casos.

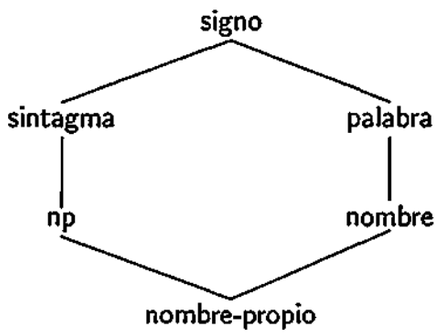


Figura 5.2: Un poset $\langle \leq, C \rangle$ con herencia múltiple

5.2.1 Los problemas de la herencia múltiple por defecto

Uno de los procesos en los que aparece con más frecuencia este tipo de problemas es el de la herencia múltiple por defecto. Cuando un tipo hereda información contradictoria de dos supertipos que no tienen relación de orden entre ellos, no es posible en general decidir cúal es la información que ese tipo debe heredar. En este caso, la solución restrictiva puede consistir en impedir a un tipo heredar valores para un atributo de más de una linea de herencia . La otra forma es proporcionando información adicional que nos permita resolver los conflictos. Un buen ejemplo son los *Órdenes de precedencia local* y las *Listas de precedencia de clases* (LPC) de CLOS. En CLOS, la especificación de los supertipos directos de un tipo se hace, en su definición, mediante una lista de superclases directas (LSD) que define el orden de precedencia local. Sobre ese orden local se construye la *Class Precedence List* o lista de precedencia de clases para ese tipo. La lista de precedencia de clases es un orden total sobre el tipo y todos sus supertipos consistente con el orden de precedencia local.

Consideremos el ejemplo de la figura 5.2.

Si existe información contradictoria entre alguna de las dos ramas de herencia para nombre-propio, no hay a priori forma de resolverla. Consideremos, sin embargo, una posible definición de esa jerarquía en CLOS:

```
(defclass signo ())
(defclass sintagma (signo))
(defclass palabra (signo))
(defclass np (sintagma))
(defclass nombre (palabra))
(defclass nombre-propio (nombre np))
```

La lista que aparece en la definición de cada tipo es su orden de precedencia local. Sobre ellas, CLOS construye las siguientes listas de precedencia de clases:

```
LPC(signo) = ()
LPC(sintagma) = (sintagma signo)
LPC(palabra) = (palabra signo)
LPC(np) = (np sintagma signo)
LPC(nombre) = (nombre palabra signo)
LPC(nombre-propio) = (nombre-propio nombre palabra np sintagma signo)
```

Disponiendo de un orden completo, dejan de existir conflictos irresolubles. Por ejemplo, si nombre-propio recibiera información contradictoria de sus supertipos palabra y np, prevalecería la proveniente de palabra, según establece la lista de precedencia de clases para nombre-propio.

Es interesante observar que no existe un único orden completo compatible con el orden local de precedencia. Incluso con la información extra que proporcionan los órdenes locales, la ordenación total es relativamente arbitraria. En particular, el algoritmo que usa CLOS para construir la lista de precedencia de clases utiliza la información de precedencia local para dar preferencia sobre líneas de herencia (en el ejemplo, nombre y todos sus supertipos tienen precedencia sobre np y sus supertipos, en cuanto a la transmisión de información a su subtipo común nombre-propio. Otra opción igualmente arbitraria hubiera sido ordenar las superclases según la "distancia" al tipo en cuestión. En ese caso, la lista de precedencia de clases para nombre-propio sería:

```
LPC(nombre-propio) = (nombre palabra np sintagma signo)
```

5.2.2 Los problemas de la unificación no monótona

Dentro de la representación computacional de conocimiento lingüístico, hay otra situación conflictiva relacionada con la herencia por defecto. Se trata de la interacción entre unificación por defecto y compartición de valores en estructuras de rasgos [Carpenter, 1993, Bouma, 1992].

Veamos un ejemplo. Tomemos las estructuras de rasgos de la figura 5.3. Si intentamos unificar las dos estructuras D y ND tomando la información de ND como más específica, hay al menos dos resultados posibles (que pueden verse en la figura 5.4.

Como en el caso de la herencia múltiple por defecto, a éste problema se le pueden dar dos tipos de soluciones: imponer restricciones o añadir información. La primera, en el caso de estructuras de rasgos, puede consistir en dejar inespecificada aquella información en la que hay conflicto. Éste es el acercamiento de

[Bouma, 1992], donde los conflictos de información se resuelven de forma *escéptica* (según expresión de [Carpenter, 1993]).

La segunda aproximación posible es la de añadir información en las estructuras de rasgos que resuelva los casos indeterminados. Ésta es la filosofía de [Lascarides et al., 1994], donde se establece un mecanismo de unificación por defecto sobre estructuras de rasgos tipadas aumentadas con especificaciones sobre la información que debe ser tomada por defecto y la que no. La operación definida en aquel trabajo tiene, además, una característica interesante frente a la Unificación por defecto de Bouma: es una operación independiente del orden. En la unificación por defecto de Bouma, la información que prevalece es la de uno de los dos argumentos, tomado como el más específico. Si se invirtieran los papeles, el resultado de la unificación por defecto sobre dos estructuras de rasgos sería diferente en cada caso. La unificación de [Lascarides et al., 1994], sin embargo, es una operación conmutativa, y la información que prevalece es aquella más específica dentro de la jerarquía del sistema.

5.3 Condiciones de buena formación sobre reglas genéricas

Nuestro problema es similar al de la herencia múltiple por defecto y al de la unificación por defecto sobre estructuras de rasgos con compartición de valores. Las posibles soluciones deben pasar por restringir el conjunto de posets cartesianos válidos, o ampliar la información que permite establecer la precedencia.

¿De qué manera podemos expresar las restricciones sobre los posets? Podemos adoptar dos posturas. Una es decidir sobre qué situaciones son erróneas o inadmisibles mientras se hace la ligadura dinámica, es decir, sobre los posets cartesianos aumentados que considera la función de ligadura dinámica G. Ésta podría llamarse "política de hechos consumados", ya que sólo se conoce la posibilidad de que haya conflictos al encontrarse con ellos. El equivalente, en un lenguaje de

$$D = \begin{bmatrix} f : a \\ g : b \end{bmatrix} \qquad ND = \begin{bmatrix} f : \boxed{1} \\ \\ g : \boxed{1} \end{bmatrix}$$

Figura 5.3: Dos estructuras de rasgos sobre las que la unificación por defecto no está determinada.

$$\begin{bmatrix} f:a \\ g:a \end{bmatrix} \qquad \begin{bmatrix} f:b \\ g:b \end{bmatrix}$$

Figura 5.4: Posibles resultados de la unificación por defecto de D y ND

programación orientado a objetos, lo encontramos en C++. Cuando una función es invocada en C++, y no existe un único método más específico que todos los demás para la tupla de argumentos en cuestión, C++ devuelve un error. La otra postura - más razonable desde nuestro punto de vista - es la de estudiar si el poset cartesiano original correspondiente a la regla genérica es susceptible de producir situaciones indeterminadas, permitiendo así el corregirlas de antemano.

Dentro de esta segunda opción, estudiaremos los conflictos de precedencia a través de *condiciones de buena formación* sobre los posets cartesianos asociados a una regla genérica. La ventaja principal es que los posets potencialmente peligrosos pueden ser detectados en tiempo de compilación, y de esa forma se puede prevenir sobre la necesidad de modificar el conjunto de reglas (en el sentido restrictivo) o añadir información sobre precedencia.

Pueden verse las condiciones de buena formación sobre la regla genérica como un mecanismo de control que compensa la libertad de la persona que la escribe. Cuando se escribe un fragmento de gramática en un formalismo de estructuras de rasgos tipadas, cada nueva pieza de información se encaja en la signatura lingüística mediante un tipo. Sin embargo, cuando se escribe una nueva regla parcial sólo hay que tener en cuenta las condiciones de tipo sobre sus argumentos, mientras que el lugar que ocupa esa regla parcial en la jerarquía de la regla genérica es deducido por el sistema. De manera que es mucho más fácil introducir comportamientos no deseados. Las condiciones de buena formación deben predecir, al menos en parte, la posibilidad de que se produzcan esos comportamientos no deseados.

¿Cúando una regla genérica es susceptible de producir conflictos irresolubles? Para ellos debe existir al menos una posible tupla de argumentos (es decir, un elemento cualquiera de $\mathcal{T} \times \mathcal{T}$) que tenga más de una superclase directa al ser introducida en la jerarquía $\langle \preceq, \mathcal{T}^* \rangle$. Para tener más de un padre ha de tener al menos dos. Y para que un tipo cartesiano $x_1 \otimes x_2$ tenga dos padres directos $a_1 \otimes a_2$ y $b_1 \otimes b_2$, no debe haber relación de orden entre éstos.

La siguiente definición proporciona un criterio que es condición necesaria y suficiente para que esa situación no pueda producirse:

Un poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle$ está **bien formado** si y sólo si
$\forall\ x_1 \otimes x_2, y_1 \otimes y_2 \in \langle \preceq, \mathcal{T}^* \rangle$ no ordenados, tales que $m_1 = x_1 \wedge y_1 \in \langle \leq, \mathcal{T} \rangle$ y $m_2 = x_2 \wedge y_2 \in \langle \leq, \mathcal{T} \rangle$, se cumple que $m_1 \otimes m_2 \in \langle \preceq, \mathcal{T}^* \rangle$

donde $x \wedge y$ representa el *máximo subtipo común* de x e y.

No está de más probar que la condición de buena formación sobre posets cartesianos es necesaria y suficiente para que no se puedan producir conflictos irresolubles al aplicar reglas genéricas.

Necesaria: Si existen dos elementos tales que $m_1 \otimes m_2 \notin \langle \preceq, \mathcal{T}^* \rangle$, entonces la tupla de argumentos m_1, m_2 presentaría herencia múltiple respecto a esos dos elementos.

Suficiente: Supongamos que, al aplicar una regla genérica bien formada sobre una tupla de argumentos con tipos x_1, x_2, nos encontramos con que el tipo introducido dinámicamente $t_1 \otimes t_2$ tiene dos padres directos $x_1 \otimes x_2$ y $y_1 \otimes y_2$. Por la definición de $\preceq$, se cumple que $t_1 \leq x_1, t_1 \leq y_1$ y que $t_2 \leq x_2, t_2 \leq y_2$. Pero entonces, necesariamente se cumple que $t_1 \leq m_1, t_2 \leq m_1$, ya que m_1 y m_2 son los máximos tipos comunes de x_1, y_1 y x_2, y_2 respectivamente. Por lo tanto, $x_1 \otimes x_2 \preceq m_1 \otimes m_2$. Pero entonces $x_1 \otimes x_2, y_1 \otimes y_2$ no son los padres directos de $t_1 \otimes t_2$, que es la hipótesis de la que partíamos.

Tanto la definición de buena formación como su prueba asumen que en el poset original $\langle \leq, \mathcal{T} \rangle$ cada par de tipos tiene, como mucho, un máximo subtipo común. Ésta condición se cumple siempre si $\langle \leq, \mathcal{T} \rangle$ es un retículo, en cuyo caso $\wedge$ es una operación interna (*intersección*) de $\langle \leq, \mathcal{T} \rangle$ y, por lo tanto, hay exactamente un elemento intersección para cada par de elementos de T. Como estamos hablando de simples posets, pueden darse casos en los que haya más de un máximo subtipo común (ver figura 5.5).

Sin embargo, siempre es posible reescribir el poset de forma que se conserven las relaciones de orden originales añadiendo un nuevo tipo. En la figura 5.5 se muestra ese proceso para un poset sencillo.

Desde un punto de vista puramente algebraico, el que un poset cartesiano no esté bien formado es algo que sucede con mucha facilidad. De hecho, cualquier poset que incluya, al menos, una relación de orden entre dos de sus miembros, admite al menos un poset cartesiano mal formado. Por ejemplo, si $\langle \leq, \mathcal{T} \rangle = \{t_1, t_2, \ldots\}, t_1 \leq t_2$, entonces el poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle = \{t_1 \otimes t_2, t_2 \otimes t_1\}$ está mal formado.

Desde el punto de vista de una gramática, sin embargo, la aparición de este tipo de patologías indica que se han dejado huecos en la gramática, interacciones que no se han tenido en cuenta. La ventaja de las condiciones de buena formación es que permiten la detección de posibles conflictos en tiempo de compilación.

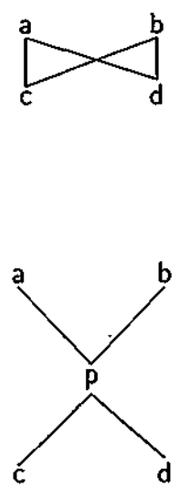


Figura 5.5: El poset de la parte superior no cumple la condición de que cada par de tipos tenga, a lo sumo, un mínimo supertipo común. Puede reescribirse en el poset de la parte inferior, que conserva las relaciones de orden del primero y si cumple esa condición.

5.4 Reparación de malformaciones

De la condición de buena formación se sigue trivialmente una forma de convertir un poset malformado en otro bien formado sin necesidad de introducir ningún mecanismo adicional de ordenamiento sobre reglas parciales. Si entre dos de ellas, $x_1 \otimes x_2, y_1 \otimes y_2$, se puede producir un conflicto, basta con introducir una nueva regla parcial cuyo tipo cartesiano sea precisamente $m_1 \otimes m_2$, donde $m_1 = x_1 \wedge y_1$ y $m_2 = x_2 \wedge y_2$. Si se quiere establecer, simplemente, que la regla que debe usarse en caso de conflicto es $x_1 \otimes x_2$ y no $y_1 \otimes y_2$, por ejemplo, basta con identificar la nueva regla parcial $m_1 \otimes m_2$ con la misma operación que $x_1 \otimes x_2$. De esta forma, se ha impuesto una ordenación adicional sin utilizar mecanismos expresivos adicionales (como es el caso de las *Listas de precedencia local* de CLOS).

Fijémonos, por ejemplo, en la figura 5.1. El poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle$ está mal formado, porque

$$\begin{array}{l} \text{posesivo} \wedge \text{pronombre} = \text{posesivo} \in \mathcal{T}^* \\ \text{nombre} \wedge \text{nombre-contable} = \text{nombre-contable} \in \mathcal{T}^* \\ \text{posesivo} \otimes \text{nombre y pronombre} \otimes \text{nombre-contable no están ordenados} \\ \text{posesivo} \otimes \text{nombre-contable} \notin \mathcal{T}^* \end{array} \tag{5.3}$$

Por lo tanto, la forma de posesivo⊗nombre-contable no pertenece a $\langle \preceq, \mathcal{T}^* \rangle$. Por eso pueden producirse situaciones indeterminadas al aplicar una regla genérica con ese poset cartesiano asociado, como al aplicarla sobre "sus gallinas". La solución pasa por introducir una nueva regla con tipo posesivo⊗nombre-contable, que puede especificarse como la misma operación que alguna de las correspondientes a posesivo⊗nombre y pronombre⊗nombre-contable.

La nueva regla genérica ya no produce indeterminaciones en casos como el de "sus gallinas" (ver figura 5.6).

5.5 Listas de precedencia de clases

La condición de buena formación puede relajarse en caso de que los elementos de la jerarquía léxica original, $\mathcal{T}$, tengan algún tipo de ordenación adicional, como las listas de precedencia de clases de CLOS.

Como hemos visto, en CLOS los casos conflictivos de herencia múltiple se resuelven por medio de órdenes locales que establecen precedencia entre los supertipos directos de cada tipo de la jerarquía. Para extender esta ordenación a los tipos cartesianos de las reglas, necesitamos introducir aquí los órdenes de prece-

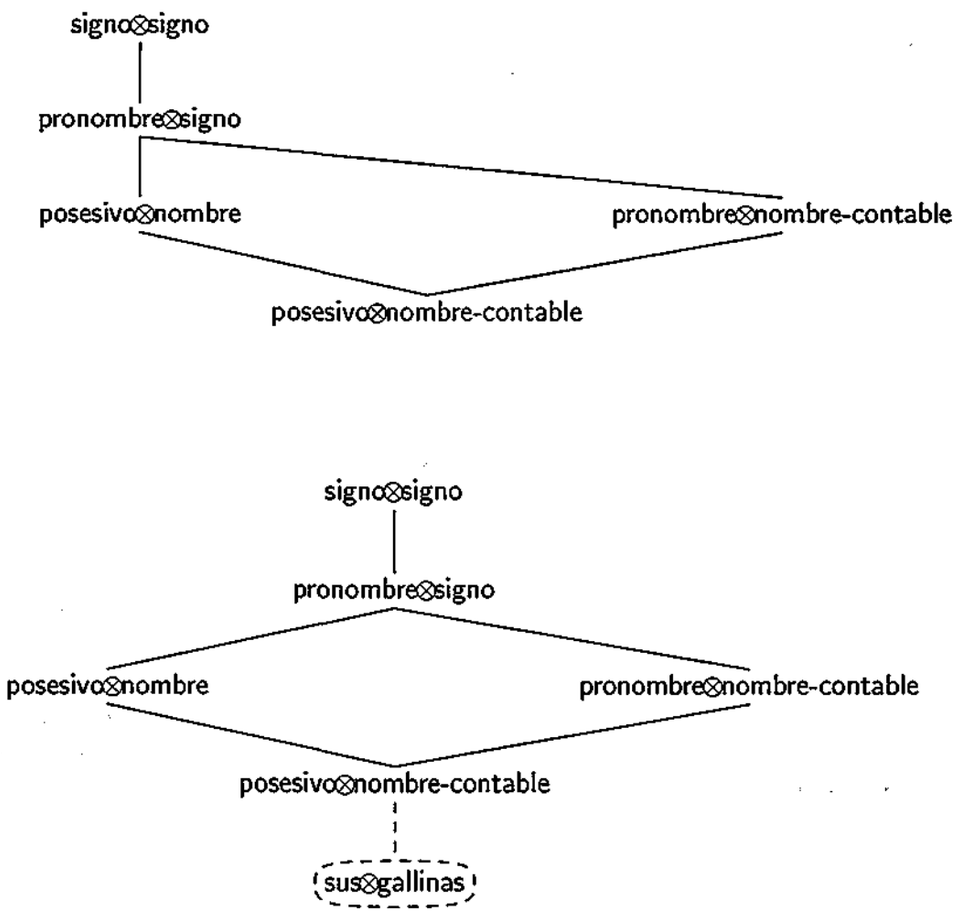


Figura 5.6: El poset cartesiano de la figura 5.1, modificado para que cumpla la condición de buena formación. Al aplicarlo ahora sobre "sus gallinas", existe una única regla más específica de tipo cartesiano posesivo⊗nombre-contable

dencia locales y las listas de precedencia de clases de forma algo más rigurosa. Para ello nos basaremos en [Bobrow et al., 1990].

Cada elemento de $\mathcal{T}$ lleva asociado una lista de superclases directas (LSD):

$$\text{LSD}(C) = (C_1, C_2, \ldots, C_n) \tag{5.4}$$

Una clase precede a todas sus superclases directas, y una superclase directa precede a todas las demás superclases directas especificadas a su derecha en la LSD. Para cada clase C se define el siguiente conjunto:

$$R_C = \{(C, C_1), (C_1, C_2), (C_2, C_3) \ldots (C_{n-1}, C_n)\} \tag{5.5}$$

Estos pares ordenados generan el ordenamiento total sobre la clase C y sus superclases directas. Si llamamos S_C al conjunto de todas las superclases de C, definimos:

$$R = \bigcup_{c \in S_C} R_c \tag{5.6}$$

El conjunto R genera un orden parcial sobre los elementos de S_C (siempre que los R_c sean consistentes). A partir de R se construye la lista de precedencia de clases para C, ordenando topológicamente los elementos de S_C con respecto al orden parcial generado por R.

El ordenamiento topológico consiste en lo siguiente: en cada paso, se añade a la lista de precedencia de clases una clase en S_C tal que ninguna otra clase la precede, de acuerdo con los elementos de R. Si hay más de una clase candidata, se escoge aquella que tiene una subclase directa más a la derecha de la lista de precedencia de clases que se lleva computada. Éste último criterio convierte el proceso de construcción de la lista de precedencia de clases en un proceso determinista.

El efecto del criterio para seleccionar de un conjunto de clases sin predecesores es que las clases en una cadena simple de superclases están adyacentes en la lista de precedencia de clases, y cada conjunto de clases en cada subgrafo separado están adyacentes en la lista de precedencia de clases.

Veamos un ejemplo. Si asignamos al poset $\langle \leq, C \rangle$ de la figura 5.2 las siguientes listas de superclases directas:

```
LSD(signo) = ()
LSD(sintagma) = (signo)
LSD(palabra) = (signo)
LSD(np) = (sintagma)
LSD(nombre) = (palabra)
LSD(nombre-propio) = (nombre np)
```

entonces los elementos que intervienen en la formación de la lista de precedencia de clases (LPC) para el tipo nombre-propio son:

$$
\begin{aligned}
&R_{\text{nombre-propio}} = \{(\text{nombre-propio}, \text{nombre}), (\text{nombre}, \text{np})\} \\
&S_{\text{nombre-propio}} = \{\text{nombre-propio}, \text{nombre}, \text{np}, \text{palabra}, \text{sintagma}, \text{signo}\} \\
&R = \{(\text{nombre-propio}, \text{nombre}), (\text{nombre}, \text{np}), (\text{nombre}, \text{palabra}), \\
&\quad (\text{palabra}, \text{signo}), (\text{np}, \text{sintagma}), (\text{sintagma}, \text{signo})\} \\
&\text{LPC(np)} = (\text{nombre-propio}, \text{nombre}, \text{palabra}, \text{np}, \text{sintagma}, \text{signo})
\end{aligned}
\tag{5.7}
$$

Ahora consideremos el poset cartesiano $\langle \preceq, \mathcal{C}^* \rangle$ formado por los tipos sintagma⊗nombre, nombre⊗nombre-propio.

Al incluir un nuevo tipo como Juan⊗Pedro (con Juan, Pedro $\in \langle \leq, \mathcal{T} \rangle$, Pedro $\preceq$ nombre-propio, Juan $\preceq$ nombre-propio) nos encontramos con herencia múltiple respecto a las dos reglas representadas por los tipos sintagma⊗nombre y nombre⊗nombre-propio. Sin embargo, podríamos hacer uso de las listas de precedencia de clases sobre el conjunto original $\langle \leq, \mathcal{T} \rangle$ para romper esta indeterminación.

Para ello introducimos la siguiente definición:

> Sea un conjunto parcialmente ordenado $\langle \leq, \mathcal{T} \rangle$ dotado con listas de precedencia de clases y un poset cartesiano asociado $\langle \preceq, \mathcal{T}^* \rangle$. Dados $x_1, x_2, y_1, y_2, a_1, a_2 \in \langle \leq, \mathcal{T} \rangle$ y $x_1 \otimes x_2, y_1 \otimes y_2 \in \langle \preceq, \mathcal{T}^* \rangle$, $a_1 \otimes a_2 \in \mathcal{T} \times \mathcal{T}$, diremos que $x_1 \otimes x_2$ **precede a** $y_1 \otimes y_2$ **con respecto a** $a_1 \otimes a_2$, y lo escribiremos
>
> $$x_1 \otimes x_2 \preceq_{a_1 \otimes a_2} y_1 \otimes y_2$$
>
> si y sólo si x_1 aparece a la izquierda de y_1 en la lista de precedencia de clases para a_1 y x_2 aparece a la izquierda de y_2 en la listas de precedencia de clases para a_2.

Si el conjunto de supertipos cartesianos directos de $a_1 \otimes a_2$ tiene ínfimo, es decir, un elemento $i_1 \otimes i_2$ tal que $\forall x_1 \otimes x_2 \in \mathcal{T}^*, i_1 \otimes i_2 \preceq_{a_1 \otimes a_2} x_1 \otimes x_2$, entonces puede utilizarse la regla parcial asociada a $i_1 \otimes i_2$ como la más específica para los argumentos a_1 y a_2.

Y entonces también podemos reescribir la condición de buena formación para posets cartesianos como sigue:

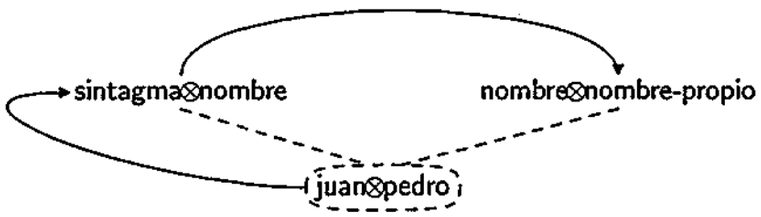


Figura 5.7: $\langle \preceq, \mathcal{C}^* \rangle \cup$Juan⊗Pedro. Aunque el orden parcial entre las reglas no señala una única regla más específica (líneas discontinuas) , la información de las listas de precedencia de clases permite resolver la indeterminación (líneas curvadas)

Un poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle$ definido sobre un poset $\langle \leq, \mathcal{T} \rangle$ equipado con listas de precedencia de clases está **bien formado** si y sólo si
$\forall\ x_1 \otimes x_2, y_1 \otimes y_2 \in \langle \preceq, \mathcal{T}^* \rangle$ no ordenados, tales que $m_1 = x_1 \wedge y_1 \in \langle \leq, \mathcal{T} \rangle$ y $m_2 = x_2 \wedge y_2 \in \langle \leq, \mathcal{T} \rangle$, se cumple que $m_1 \otimes m_2 \in \langle \preceq, \mathcal{T}^* \rangle$ o que el conjunto de superclases directas de $m_1 \otimes m_2$ en $\mathcal{T}^* \cup \{m_1 \otimes m_2\}$ tiene un ínfimo conrespecto a la relación de orden $\preceq_{m_1 \otimes m_2}$

Para introducir ésta ordenación extra en el comportamiento de las reglas genéricas, sólo hay que reescribir la función de ligadura dinámica de ésta forma:

$\forall x_1, x_2 \in \mathcal{T}, G(x_1, x_2) \equiv f_{t_1 \otimes t_2}$ tal que $t_1 \otimes t_2$ es la mínima cota superior de $x_1 \otimes x_2$ en $\mathcal{T}^* \cup \{x_1 \otimes x_2\}$. Si no existe cota superior, se escoge la regla $f_{i_1 \otimes i_2}$ tal que $i_1 \otimes i_2$ es el ínfimo del conjunto de supertipos directos de $x_1 \otimes x_2$ en $\mathcal{T}^* \cup \{x_1 \otimes x_2\}$ con respecto al orden $\preceq_{x_1 \otimes x_2}$.

Al introducir el tipo Juan⊗Pedro en el poset cartesiano formado por los tipos sintagma⊗nombre, *ctypenombrenombre − propio*, se obtiene la jerarquía extendida de la figura 5.7.

Si no dispusiéramos de listas de precedencia de clases, no podríamos establecer cuál sería el comportamiento de una dotada con el poset cartesiano $\langle \preceq, \mathcal{C}^* \rangle$ al ser invocada sobre la tupla de argumentos Juan, Pedro. Sin embargo, los órdenes locales definidos anteriormente resuelven el conflicto. Efectivamente: el conjunto de

supertipos directos de Juan⊗Pedro es {sintagma⊗nombre, nombre⊗nombre-propio}. Como LSD(Juan) = (nombre-propio) y LSD(Pedro) = (nombre-propio), las listas de precedencia de clases correspondientes son:

$$\begin{array}{l}\text{LPC(Juan)} = \text{(nombre-propio nombre palabra np sintagma signo)} \\ \text{LPC(Pedro)} = \text{(nombre-propio nombre palabra np sintagma signo)}\end{array} \tag{5.8}$$

Por lo tanto, se cumple que nombre⊗nombre-propio $\preceq_{\text{Juan⊗Pedro}}$ sintagma⊗nombre, y nuestra segunda definición para la función de ligadura dinámica es capaz de establecer la precedencia entre las reglas parciales asociadas.

Debe quedar claro que esta extensión a la capacidad del sistema para ordenar reglas parciales no quiere decir, ni mucho menos, que todos los posets cartesianos vayan a estar bien formados si la jerarquía léxica original produce ordenamientos totales para las superclases de cada clase dada. El hecho de que la precedencia, en los posets cartesianos, se produzca por la intersección de dos dimensiones de ordenación, hace que ningún ordenamiento sobre $\mathcal{T}$ sea suficiente para garantizar la ordenabilidad en $\mathcal{T}^*$. Salvo, claro está, en el caso trivial en el que la jerarquía original $\langle \leq, \mathcal{T} \rangle$ no contenga ninguna relación de orden, es decir, su estructuración sea equivalente a un conjunto de no-terminales en una gramática libre de contexto tradicional. Si hay relaciones de orden, aunque sólo sea una, es posible encontrar posets cartesianos malformados. Efectivamente, si consideramos el caso más simple:

$$\langle \leq, \mathcal{T} \rangle = \{t_1, t_2, \ldots\}, t_1 \leq t_2 \tag{5.9}$$

entonces el siguiente poset cartesiano está malformado, independientemente de cúales sean los posibles órdenes locales definidos sobre los elementos de $\langle \leq, \mathcal{T} \rangle$:

$$\langle \preceq, \mathcal{T}^* \rangle = \{t_1 \otimes t_2, t_2 \otimes t_1\} \tag{5.10}$$

Llegados a este punto, no se puede flexibilizar más el criterio de buena formación sin violar el principio fundamental por el que operan las reglas genéricas: que una regla parcial es más específica que otra si lo son los argumentos de una con respecto a los de la otra. Si no se respeta este principio, creemos que las reglas genéricas perderían sentido lingüístico, y su funcionamiento desafiaría la intuición.

Comparemos nuestros criterios de precedencia con los de CLOS. En CLOS se llama *método efectivo* para una tupla de argumentos de una función genérica al código que se ejecuta sobre estos argumentos cuando se invoca la función genérica. El método efectivo es una combinación de los métodos aplicables en la función

genérica. Cuando se ha determinado cúal es el método efectivo, éste es invocado con los mismos argumentos que se pasaron a la función genérica. El método efectivo se determina en tres pasos:

1. Seleccionar los métodos aplicables.
2. Ordenarlos por orden de precedencia, poniendo primero el método más específico.
3. Aplicar la combinación de métodos a la lista ordenada de métodos aplicables, produciendo el método efectivo.

El primer paso es trivial y no vamos a detenernos en él. El último, dependiendo de los distintos tipos de métodos(*primary, around, before, after*, etc.), es un proceso complicado y modificable por el programador, en el que tampoco vamos a detenernos. Nos interesa ver cúal es el mecanismo de ordenación. Éste es el siguiente: para comparar la precedencia de dos métodos, sus *especializadores de parámetro*, que vienen a ser los tipos de sus argumentos, se examinan en orden. El orden en que se examinan es siempre de izquierda a derecha. La diferencia entre especializadores de parámetro y tipos estriba en que, en CLOS, la especificación sobre cada argumento puede no ser un tipo. Por ejemplo, un especializador de parámetro válido puede ser `(eql Juan)`, que quiere decir que ese argumento ha de ser exactamente el objeto Juan. Esta distinción es necesaria en lenguajes que, como CLOS, distinguen entre *clases* e *instancias* de las clases. No lo es, sin embargo, en una aproximación basada en *prototipos*, en la que no hay distinción entre clases y objetos, como la que se utiliza en Loom [Mcgregor, 1990] y en casi todos los sistemas de representación de conocimiento lingüístico, como ALE, CUF, TFS, etc. Nosotros hemos hablado de tipos siempre entendiéndolos de esta segunda forma. se examinan en orden.

Empezando, pues, por la izquierda, se comparan los especializadores de parámetro. Cuando un par de especializadores son iguales, se compara el par siguiente. Si no son iguales, el más específico de los dos métodos es aquel cuyo especializador de parámetro aparece antes en la lista de precedencia de clases del argumento correspondiente. Recordemos que la lista de precedencia de clases es un orden total sobre las superclases de una clase dada, así que los especializadores han de aparecer necesariamente en esa lista.

Gracias a la lista de clases y al procedimiento asimétrico mediante el que se establece precedencia, CLOS siempre es capaz de dar una ordenación total para los métodos aplicables en cualquier situación dada. No hay ninguna restricción sobre los posibles métodos comparable a nuestro criterio de buena formación, porque no hay situaciones que el sistema no sepa resolver.

Por el contrario, en una regla genérica sólo se puede garantizar una ordenación parcial sobre las reglas aplicables, y una única regla más específica que el resto.

El sistema de CLOS, aplicado a nuestras reglas parciales, no tendría ningún sentido lingüistico. No hay ninguna razón por la cual la especificidad de una regla debiera hacerse en torno al objeto lingüístico que aparece antes en la oración. El hecho de que la función de ligadura dinámica fuese capaz de manejar cualquier tipo de acumulación de reglas parciales excedería la potencia de las reglas genéricas y oscurecería su funcionamiento.

La posibilidad de obtener una ordenación total sobre las reglas aplicables, sin embargo, sí es interesante desde el punto de vista de las reglas sintagmáticas, pues permitiría combinaciones más sofisticadas de las reglas aplicables que la simple elección de la regla más específica. Por ejemplo, en el caso de que las reglas se especificaran mediante estructuras de rasgos y se unificaran mediante algún tipo de unificación por defecto, una ordenación total sobre las reglas aplicables impediría que surgieran contradicciones entre reglas con la misma precedencia, al no existir reglas con la misma precedencia. Afortunadamente, veremos en el próximo capítulo que el uso de la regla más específica como regla efectiva resulta suficiente para dar cuenta de los procesos de interpretación semántica y sintáctica. De hecho, la identificación entre método más específico y código efectivo es la mas usada en programación orientada a objetos. Hemos citado en los últimos párrafos el caso de CLOS; en el caso de C++, que es quizás el lenguaje más usado de los orientados a objetos, se produce exactamente la misma identificación. A diferencia de nuestra aproximación, sin embargo, el procedimiento de selección es algorítmico y no algebraico, y sólo se detectan posibles malas formaciones cuando se produce un error en tiempo de ejecución.

En cualquier caso, creemos que nuestra aproximación es más razonable para establecer precendencia entre reglas lingüísticas por los siguientes motivos:

1. En primer lugar, la precedencia no se establece de forma algorítmica más o menos arbitraria, sino algebraicamente mediante un conjunto parcialmente ordenado.

2. En segundo lugar, las condiciones de buena formación no reducen la expresividad, sino que obligan a realizar especificaciones adicionales en los puntos en los que la gramática no está suficientemente definida.

3. En tercer lugar, la buena formación puede ser comprobada automáticamente en tiempo de compilación, y en caso de que un poset esté malformado, el sistema señala automáticamente el tipo cartesiano de la regla parcial que hay que incluir en el sistema. Ésta es la forma más cómoda de asegurar la coherencia de una regla genérica.

Capítulo 6

Procesamiento con reglas genéricas

En este capítulo estudiamos los aspectos esenciales del procesamiento de reglas genéricas. En primer lugar, damos varios algoritmos de parsing para gramáticas parcial o totalmente descritas mediante reglas genéricas. Para ello, adaptamos técnicas de parsing de gramáticas libres de contexto a las características de nuestro formalismo. A pesar de ser éste último no monótono, el coste computacional asociado es muy parecido al de las gramáticas libres de contexto. En particular, la dependencia con el tamaño de la cadena es la misma.

En segundo lugar, especificamos un algoritmo de compilación de reglas genéricas en sistemas monótonos de reglas sintagmáticas. Este algoritmo a) hace explícita la relación entre reglas genéricas y gramáticas sintagmáticas, b) permite considerar la posibilidad de extender sistemas ya existentes con la capacidad de interpretar reglas por defecto, y c) completa los razonamientos sobre coste.

6.1 Parsing

Es interesante considerar cada una de las siguientes posibilidades en la descripción de una gramática:

1. La gramática consiste en una regla genérica que combina información sintáctica. Éste es el caso de la aplicación que se presenta en el capítulo 7.

2. Se dispone de una gramática libre de contexto (posiblemente aumentada) que invoca a una regla genérica para realizar la combinación semántica. Este arreglo permite una descripción adecuada de la interacción entre un sistema categorial de interpretación semántica y una gramática libre de contexto, soslayando las dificultades de los formalismos de Unificación para representar adecuadamente el alcance de los cuantificadores.

3. La gramática consiste en una regla genérica que combina información sintáctica y otra que combina información semántica. El sistema que se describe en el capítulo 8 se ajusta a esta descripción.

Partiremos de un algoritmo para reglas libres de contexto y lo modificaremos para reconocer reglas genéricas, según cada uno de estos tres casos.

6.1.1 Interacción de una regla genérica con una gramática libre de contexto

La forma más sencilla de llegar a un algoritmo de parsing para reglas genéricas es considerando primero el caso 2, en el que una gramática libre de contexto invoca a una regla genérica para realizar la interpretación semántica. Para ello partimos del caso, bien conocido, en que una gramática libre de contexto realiza la interpretación semántica mediante una función de interpretación semántica asociada a cada regla libre de contexto. La disposición funcional del conocimiento sobre interpretación semántica minimiza el salto conceptual a la descripción de ese proceso mediante una regla genérica.

El estudiar el parsing de gramáticas libres de contexto no quiere decir que los resultados no sean aplicables para gramáticas de unificación. En la primera parte de la memoria vimos que este tipo de gramáticas tiene asociado (explícita o implícitamente) un componente libre de contexto sobre el que se aplican las técnicas de parsing usuales. Los procesos de Unificación asociados a la aplicación de cada regla involucran, eso sí, un factor de complejidad asociado que hay que tener en cuenta en cada caso [1].

Para estudiar los algoritmos utilizaremos la notación de sistemas deductivos de [Shieber et al., 1994]. Aunque el carácter algorítmico de estos sistemas no es tan explícito como otras notaciones más convencionales, su simplicidad resalta los aspectos cruciales de los procesos de parsing y, en particular, facilita la visualización de las manipulaciones que llevaremos a cabo sobre algoritmos usuales para que procesen información asociada a reglas genéricas.

En [Shieber et al., 1994], los procesos de parsing se entienden como procesos deductivos en los que se usan reglas de inferencia para derivar hechos acerca de la gramaticalidad de cadenas a partir de otros hechos. La forma general de una regla de inferencia es:

$$\frac{A_1 \quad \ldots \quad A_k}{B} \quad \langle \text{condiciones laterales sobre } A_1, \ldots, A_k, B \rangle \quad .$$

[1] La interacción entre reglas libres de contexto y restricciones funcionales, y el coste computacional asociado, se discuten en [Maxwell and Kaplan, 1994].

donde antecedentes y consecuente son esquemas de fórmulas, es decir, pueden incluir metavariables sintácticas que han de ser sustituidas por términos adecuados al aplicar la fórmula. La derivación de una fórmula B a partir de unas hipótesis $A_1 \quad \ldots \quad A_k$ es una secuencia de fórmulas tales que B es la última, y las demás son axiomas o se pueden obtener a través de la aplicación de una de las reglas de inferencia. Escribimos $A_1, \ldots, A_m \vdash B$ si existe tal derivación, y decimos que B es una consecuencia de $A_1 \quad \ldots \quad A_k$.

Las condiciones laterales pueden hacer referencia a las reglas de una gramática concreta, y las fórmulas pueden referirse a posiciones en la cadena $w = w_1 w_2 \ldots w_n$ que va a ser analizada.

Queremos analizar una cadena $w = w_1 w_2 \ldots w_n$ con una gramática libre de contexto $G = \langle N, \Sigma, P, S \rangle$, donde N es el conjunto de no terminales incluyendo el símbolo de partida, S, Σ es el conjunto de símbolos terminales, $V = N \cup \Sigma$ y P es el conjunto de producciones de la forma $A \to BC$ o $A \to a$. Nótese que expresamos la gramática en forma normal de Chomsky, que es la forma adecuada para interactuar con las reglas genéricas.

Consideramos una lógica con elementos de la forma $[\alpha \bullet, j]$. Un elemento $[\alpha \bullet, j]$ equivale a la afirmación, en términos de la gramática libre de contexto:

$$\alpha w_{j+1} \ldots w_n \stackrel{*}{\Rightarrow} w_1 \ldots w_n$$

Partiremos del siguiente sistema deductivo, presentado en [Shieber et al., 1994]:

Parser shift-reduce

Axiomas:	$[\bullet, 0]$	
Objetivo:	$[S \bullet, n]$	
Reglas de inferencia:		
Shift	$\dfrac{[\alpha \bullet, j]}{[\alpha w_{j+1} \bullet, j+1]}$	(6.1)
Reduce	$\dfrac{[\alpha \gamma \bullet, j]}{[\alpha B \bullet, j]} \quad B \to \gamma$	

El axioma $[\bullet, 0]$ equivale a $w_1 \ldots w_n \stackrel{*}{\Rightarrow} w_1 \ldots w_n$. El objetivo $[S \bullet, n]$ equivale a $S \stackrel{*}{\Rightarrow} w_1 \ldots w_n$. Los elementos a la derecha del punto '$\bullet$' son los que no han sido todavía considerados en el análisis. La regla de 'shift' pasa el primero de esos elementos a la izquierda del punto, y la regla de 'reduce' aplica una regla de la

gramática sobre los elementos (terminales y no terminales) inmediatamente a la izquierda del punto.

Para acercarnos gradualmente al problema de parsing que queremos describir, comenzaremos por recordar como puede aumentarse esta especificación para asociar un proceso de interpretación semántica al parsing puramente sintáctico.

Consideremos la siguiente reformulación de los elementos de la lógica de 6.1:

$$[\alpha \bullet, j, \Phi] \tag{6.2}$$

El nuevo término Φ es una expresión semántica, o cadena de expresiones semánticas, asociada a los elementos de α. De modo que, si utilizamos la notación $\alpha_i : \phi_i$ para significar el símbolo α_i con expresión semántica asociada ϕ_i, la expresión $[\alpha \bullet, j, \Phi]$ es equivalente a

$$\alpha : \Phi w_{j+1} : \phi_{j+1} \ldots w_n : \phi_n \stackrel{*}{\Rightarrow} w_1 : \phi_1 \ldots w_n : \phi_n \tag{6.3}$$

Asumiremos que las reglas de la gramática son ahora de esta forma:

$$B \rightarrow \gamma, \quad f \tag{6.4}$$

donde f es la función de composición semántica asociada a la regla libre de contexto $B \rightarrow \gamma$.

El cálculo 6.1 puede entonces reescribirse de la siguiente forma:

Parser con interpretación semántica

Axiomas: $[\bullet, 0, \ ]$
Objetivo: $[S \bullet, n, \phi_S]$
Reglas de inferencia:

Shift
$$\frac{[\alpha \bullet, j, \Phi]}{[\alpha w_{j+1} \bullet, j+1, \Phi \quad \phi_{j+1}]} \tag{6.5}$$

Reduce
$$\frac{[\alpha \gamma \bullet, j, \Phi < \phi_1 \ldots \phi_{|\gamma|} >]}{[\alpha B \bullet, j, \Phi < f(\phi_1 \ldots \phi_{|\gamma|}) >]} \quad B \rightarrow \gamma, \mathrm{f}$$

Veamos un ejemplo . Consideremos la siguiente gramática:

$$\begin{array}{ll} \mathsf{S} \rightarrow \mathsf{NP\ VP} & \lambda x_{np} \lambda y_{vp}.y_{vp}(x_{np}) \\ \mathsf{VP} \rightarrow \text{llora} & \lambda x.x \\ \mathsf{NP} \rightarrow \text{Pepe} & \lambda x.x \end{array} \tag{6.6}$$

La oración "Pepe llora", a la sazón la única permitida por tan paupérrima gramática, podría ser obtenida de esta forma:

1. [•, 0,]
2. [Pepe•, 1, *pepe*] (shift)
3. [NP•, 1, *pepe*] (reduce)
4. [NP llora•, 2, *pepe llora*] (shift)
5. [NP VP •, 2, *pepe llora*] (reduce)
6. [S•, 2, *llora(pepe)*] (reduce)

Podemos aumentar el parser de forma parecida para combinar la información asociada a una regla genérica. Para ello debemos tener en cuenta dos diferencias con respecto al caso anterior, en el que tenemos una regla de combinación semántica asociada a cada regla libre de contexto. Son éstas:

1. La acción de la regla genérica debe especificarse a través de la función de ligadura dinámica. No hay una regla parcial asociada, de antemano, a cada regla libre de contexto. Luego comprobaremos porqué esa asociación no es posible a priori.
2. Hay que modificar ligeramente la interpretación de las reglas para dar cuenta de la jerarquización de los no terminales. Una jerarquía es equivalente, en términos de una CFG, a una serie de reglas unarias en las que se especifica cada relación de orden inmediato escribiendo el supertipo a la izquierda y el subtipo a la derecha. Es decir, si tenemos que $A \leq B$ en la jerarquía, sería equivalente a la regla $B \rightarrow A$. Al estar este tipo de reglas absorbidas por la jerarquía, cuando en la gramática libre de contexto escribimos $T \rightarrow T_1 T_2$, su interpretación es la siguiente: $T \rightarrow X_1 X_2$ tales que $X_1 \leq T_1, X_2 \leq T_2$.

El elemento básico de la lógica sigue siendo el mismo que en el caso de la gramática libre de contexto aumentada con interpretación semántica: $[\alpha \bullet, j, \Phi]$. Ahora Φ es una expresión que pertenece al dominio informacional asociado a una función genérica $\mathcal{G} = \langle\langle \preceq, \mathcal{T}^* \rangle, \mathcal{F}, \mathrm{G}\rangle$ donde $\mathcal{F}$ es el conjunto de reglas parciales, $\langle \preceq, \mathcal{T}^* \rangle$ su poset cartesiano asociado y G la función de ligadura dinámica.

El sistema deductivo que da cuenta de la interacción de una regla genérica con una gramática libre de contexto es:

Parser CFG + regla genérica

Axiomas: $[\bullet, 0, \ ]$

Objetivo: $[S \bullet, n, \phi_S]$

Reglas de inferencia:

Shift

$$\frac{[\alpha \bullet, j, \Phi]}{[\alpha w_{j+1} \bullet, j+1, \Phi \phi_{j+1}]} \tag{6.7}$$

Reduce

$$\frac{[\alpha \mathsf{x}_1 \mathsf{x}_2 \bullet, j, \Phi < \phi_1 \phi_2 >]}{[\alpha A \bullet, j, \Phi < \mathrm{G}(\mathsf{x}_1, \mathsf{x}_2)(\phi_1, \phi_2) >]} \quad \begin{array}{l} A \rightarrow BC, \mathsf{x}_1 \leq B, \mathsf{x}_2 \leq C \\ \mathrm{G}(\mathsf{x}_1, \mathsf{x}_2)(\phi_1, \phi_2) \neq \perp \end{array}$$

La única regla diferente es la regla de reducción. La interacción con la regla genérica se obtiene mediante el término $\mathrm{G}(\mathsf{x}_1, \mathsf{x}_2)(\phi_1, \phi_2)$. La parte izquierda del término, $\mathrm{G}(\mathsf{x}_1, \mathsf{x}_2)$, efectúa la ligadura dinámica, devolviendo la regla efectiva que se aplicará sobre $\mathsf{x}_1, \mathsf{x}_2$.

Las condiciones de contorno para la aplicación de la regla son: que exista una regla sintáctica aplicable $(A \rightarrow BC, \mathsf{x}_1 \leq B, \mathsf{x}_2 \leq C)$, y que la función genérica, aplicada sobre ϕ_1, ϕ_2, devuelva un resultado no nulo. Recordemos que podría darse el caso de que se cumplan las restricciones estructurales pero no las restricciones adicionales impuestas por la regla genérica. Si la regla genérica asocia expresiones semánticas, por ejemplo, podría desambiguar procesos de asignación de complementos preposicionales por criterios semánticos [2] .

Es interesante observar que el proceso de ligadura dinámica depende de los objetos en particular que intervienen en la regla de reducción, y no sólo de las condiciones de la regla libre de contexto. Esto impide considerar una "función efectiva"que se adjuntara a cada regla libre de contexto en tiempo de compilación, de forma que el proceso de ligadura dinámica quedara resuelto de antemano y no hubiera que aplicarlo durante el parsing. Este hecho se refleja en el término $\mathrm{G}(\mathsf{x}_1, \mathsf{x}_2)$. Si fuera $\mathrm{G}(\mathsf{B}, \mathsf{C})$, la ligadura podría hacerse en tiempo de compilación.

[2] Como en la oración "Ví un globo rojo volando", en la que "volando" se refiere necesariamente al globo, siempre que el hablante no sea un paracaidista.

Este comportamiento es precisamente el que permite especificar la interpretación semántica y la sintáctica como procesos independientes que interaccionan modularmente entre sí.

Respecto al coste del proceso de parsing, la única diferencia con respecto a una CFG se produce en la regla de reducción. Para aplicarla hay que realizar dos tareas: 1) Buscar alguna regla sintáctica aplicable dentro del conjunto de reglas sintagmáticas de la forma $A \rightarrow BC$, y 2) Invocar la regla genérica $\mathcal{G}$ mediante la función de ligadura dinámica $\mathbf{G}$. La primera involucra, en el peor de los casos, una comparación de los elementos que se combinan con las partes derechas de cada una de las reglas de la gramática, y es necesaria también para analizar una CFG. En la segunda, la llamada a la función de ligadura dinámica involucra la introducción del tipo cartesiano $x_1 \otimes x_2$ en el el poset cartesiano correspondiente a la regla genérica. Para ello, en el peor de los casos, $x_1 \otimes x_2$ tiene que ser comparado con cada uno de los miembros de ese poset. Por lo tanto, tenemos un factor lineal añadido de complejidad en el número de reglas parciales $|\mathcal{F}^*|$que componen la regla genérica. El número de veces que se aplica la regla de reducción es exactamente la misma que con cualquiera de los dos algoritmos de parsing anteriores, de modo que la dependencia del coste con el tamaño de la gramática es la misma que para una gramática libre de contexto. Otra forma de ver este hecho es recordando la exposición que hicimos del algoritmo CYK en el capítulo 2. Tanto la tabla de $n \times n$ como la forma de rellenar cada elemento (de coste lineal en n) siguen siendo válidas; el único cambio está en la forma de invocar la gramática en cada uno de los $\mathcal{O}(n^3)$ pasos. Por supuesto, para cada regla genérica en concreto hay que añadir el factor de complejidad inducido por la naturaleza de cada regla parcial.

Examinando el proceso en más detalle, cada comparación entre dos tipos cartesianos (de las que hacemos $\mathcal{O}(|\mathcal{F}|)$) involucra la verificación de dos dependencias jerarquícas entre los miembros de $\mathcal{T}$. La verificación de una dependencia jerárquica es equivalente al problema de encontrar un camino entre dos nodos de un grafo acíclico orientado, que tiene un coste lineal con el número de aristas (o relaciones jerárquicas directas) en $\mathcal{T}$. En el peor de los casos, el número de aristas depende cuadráticamente del número de nodos (o elementos de la jerarquía) $|\mathcal{T}|$. De modo que el coste del parsing asociado a una regla genérica es sensible también al tamaño y complejidad de la jerarquía léxica, más aún que al tamaño de la jerarquía de reglas parciales. Nótese que esta dependencia en el tamaño de la jerarquía léxica no proviene del comportamiento no monótono de las reglas genéricas, sino de que el dominio de objetos lingüísticos esté jerarquizado. Cualquier sistema monótono de reglas en el que éstas se especifiquen sobre tipos de una jerarquía tendrá el mismo factor de complejidad en el tamaño de esa jerarquía.

6.1.2 Parsing sintáctico mediante reglas genéricas

Un caso particular de regla genérica es aquel en el que las funciones de combinación devuelven simplemente un nuevo tipo. En una regla genérica así, el papel

de cada regla parcial es parecido al de una regla libre de contexto, y la regla genérica actúa como una gramática libre de contexto en la que las reglas tienen una interpretación por defecto. Este es precisamente el caso del primer ejemplo que hemos introducido en el capítulo 4.

Para este tipo de reglas genéricas, el parsing no difiere mucho del de gramáticas libres de contexto. La principal diferencia es que, debido a la interpretación funcional no monótona que se da a las reglas parciales, el tipo de procesamiento para éstas ha de ser de abajo arriba, como es el caso del algoritmo shift-reduce o el CYK.

Vamos, entonces, a especificar un algoritmo de abajo arriba para reconocer cadenas con una regla genérica sintáctica. Para ello, adaptamos de nuevo el parser de 6.1:

Parser de regla genérica sintáctica

Axiomas:	$[\bullet, 0]$	
Objetivo:	$[\mathsf{x}\bullet, n] \quad \mathsf{x} \leq \mathsf{S}$	
Reglas de inferencia:		
Shift	$\dfrac{[\alpha\bullet, j]}{[\alpha w_{j+1}\bullet, j+1]}$	(6.8)
Reduce		
$\dfrac{[\alpha \mathsf{x}_1\mathsf{x}_2\bullet, j]}{[\alpha \mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2)\bullet, j]}$	$\mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2) \neq \bot$	

Puede parecer extraña la forma de la expresión $\mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2)$, con los argumentos repetidos. Sin embargo, juegan papeles muy distintos. Los dos primeros intervienen en esa expresión como tipos de los argumentos, a partir de los cuáles se realiza el proceso de ligadura dinámica. Los dos siguientes son expresiones relacionadas con el dominio informacional sobre el que opera la regla genérica $\mathcal{SYN}$, que en este caso coincide con el de los tipos de los objetos lingüísticos.

La condición lateral, en este caso, es precisamente la que licencia la operación de combinación sintáctica. En caso de que $\mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2)$ no devuelva un resultado positivo, no se puede formar un sintagma a partir de $\mathsf{x}_1,\mathsf{x}_2$.

El parsing de este tipo de regla genérica tiene un coste totalmente equivalente al de una gramática libre de contexto. La regla reduce (la única diferente) involucra solamente una llamada a la regla genérica, que supone la inclusión de un nuevo tipo en el poset cartesiano. De nuevo, esta operación tiene un coste, a lo sumo, lineal en el número de reglas parciales $|\mathcal{F}|$. Es comparable al proceso de encontrar la (o las) reglas aplicables en una CFG, que es también lineal en el número de reglas libres de contexto. La única diferencia es que, como en el algoritmo anterior, el

coste depende también del tamaño y complejidad de la jerarquía léxica $\mathcal{T}$, siendo esa dependencia cuadrática con $|\mathcal{T}|$ en el peor de los casos.

6.1.3 Interacción entre sintaxis y semántica expresada mediante reglas genéricas

Un caso particular muy interesante es aquel en el que se dispone de una regla genérica para especificar restricciones sintagmáticas y otra para llevar a cabo la interpretación semántica. Establecer un algoritmo de parsing para este caso es muy sencillo, a partir de los ya enunciados.

En este caso, los elementos de la lógica que representa el parser de parsing shift-reduce son de la forma $[\alpha \bullet, j, \Phi]$, y disponemos de dos reglas genéricas: $\mathcal{SYN} = \langle\langle \preceq, \mathcal{T}^*_{syn}\rangle, \mathcal{F}_{syn}, \mathrm{SYN}\rangle$, definida sobre $\langle\langle \leq, \mathcal{T}\rangle, \mathcal{O}, \mathcal{T}\rangle$ y $\mathcal{SEM} = \langle\langle \preceq, \mathcal{T}^*_{sem}\rangle, \mathcal{F}_{sem}, \mathbf{SEM}\rangle$ definido sobre $\langle\langle \leq, \mathcal{T}\rangle, \mathcal{O}, \Phi\rangle$, donde Φ es el dominio informacional de expresiones semánticas.

El parser puede definirse así:

Parser regla genérica sintáctica + regla genérica semántica

Axiomas: $[\bullet, 0, \ ]$

Objetivo: $[\mathsf{x} \bullet, n, \phi_x] \quad \mathsf{x} \leq \mathsf{S}$

Reglas de inferencia:

Shift

$$\frac{[\alpha \bullet, j, \Phi]}{[\alpha w_{j+1} \bullet, j+1, \Phi\phi_{j+1}]}$$

Reduce

$$\frac{[\alpha \mathsf{x}_1\mathsf{x}_2 \bullet, j, \Phi\phi_1\phi_2]}{[\alpha \mathrm{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2) \bullet, j, \mathrm{SEM}(\mathsf{x}_1,\mathsf{x}_2)(\phi_1,\phi_2)]} \begin{array}{l}\mathrm{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2) \neq \bot \\ \mathrm{SEM}(\mathsf{x}_1,\mathsf{x}_2)(\phi_1,\phi_2) \neq \bot\end{array} \tag{6.9}$$

El coste asociado a este parser es el mismo que el anterior, pero ahora el factor lineal en el tamaño de la gramática es $\mathcal{O}(|\mathcal{F}_{syn}| + |\mathcal{F}_{sem}|)$.

Podemos usar el ejemplo 4.8 para ver el funcionamiento de este algoritmo. Allí disponíamos de las siguientes reglas genéricas:

$$\mathcal{SYN} = \begin{cases} SYN_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = \mathsf{S} \\ SYN_{\mathsf{NP}\otimes\mathsf{VP}_i}(\mathsf{X},\mathsf{Y}) = \mathsf{S}_i \end{cases} \tag{6.10}$$

$$\mathcal{SEM} = \begin{cases} SEM_{\mathsf{NP}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = sem(\mathsf{X})(sem(\mathsf{Y})) \\ SEM_{\mathsf{Proper\text{-}Noun}\otimes\mathsf{VP}}(\mathsf{X},\mathsf{Y}) = sem(\mathsf{Y})(sem(\mathsf{X})) \wedge (\lambda P.P(\mathsf{Alberto}))(sem(\mathsf{Y})) \end{cases} \tag{6.11}$$

Si consideramos formados los siguientes sintagmas:

- Ana : $\lambda P.P(\mathsf{Ana})$; Ana $\leq$ proper-noun
- se+enfadó : $\lambda x.\mathsf{ENFADADO}(x)$; se+enfadó $\leq \mathsf{VP}_i$

la aplicación del sistema a la cadena "Ana se+enfadó"sería la siguiente:

1. $[\bullet, 0,]$
2. $[\mathsf{Ana} \bullet, 1, \lambda P.P(\mathsf{Ana})]$ **(Shift)**
3. $[\mathsf{Ana\ se+enfadó} \bullet, 2, \lambda P.P(\mathsf{Ana}), \lambda x.\mathsf{ENFADADO}(x)]$ **(Shift)**
4. $[\mathsf{S}_i \bullet, 2, \mathsf{ENFADADO(Ana)} \wedge \mathsf{ENFADADO(Alberto)}]$ **(Reduce)**

Donde, en la única aplicación de la regla de reducción han intervenido los siguientes factores:

$$\mathbf{SYN}(\mathsf{Ana, se+enfado})(\mathsf{Ana, se+enfado}) = SYN_{\mathsf{NP}\otimes\mathsf{VP}_i}(\mathsf{Ana, se+enfado}) = \mathsf{S}_i$$

$$\mathbf{SEM}(\mathsf{Ana, se+enfado})(\lambda P.P(\mathsf{Ana}, \lambda x.\mathsf{ENFADADO}(x)) =$$
$$SEM_{\mathsf{Proper\text{-}Noun}\otimes\mathsf{VP}}(\lambda P.P(\mathsf{Ana}, \lambda x.\mathsf{ENFADADO}(x)) =$$
$$\mathsf{ENFADADO(Ana)} \wedge \mathsf{ENFADADO(Alberto)}$$

6.2 Compilación de reglas genéricas en gramáticas sintagmáticas

Un algoritmo que convierta reglas genéricas en gramáticas sintagmáticas abre la posibilidad de especificar información genérica y por defecto en los formalismos de bajo nivel que permiten la especificación jerarquizada del dominio de objetos lingüísticos, como ALE, CUF, Troll o TFS. El usuario podría escribir reglas con las asignaciones de tipo oportunas que incluyeran descripciones de objetos y relaciones entre ellos en el sistema elegido, y el mecanismo de compilación las convertiría en reglas de una gramática aceptable por el sistema.

Por otro lado, el disponer de un algoritmo de compilación en gramáticas libres de contexto nos aportará datos sobre la naturaleza de las reglas genéricas como mecanismo computacional, así como sobre la complejidad de los procesos de parsing asociados. En particular, nos demuestra que el parsing de reglas genéricas puede hacerse en tiempo equivalente al de gramáticas libres de contexto, al coste de un tiempo de compilación determinado.

6.2.1 Equivalencia con una gramática sintagmática en el límite de jerarquización nula

El funcionamiento jerárquico de las reglas parciales en una regla genérica depende crucialmente de que el conjunto de tipos $\mathcal{T}$ que describe a los objetos lingüísticos esté jerarquizado. La estrecha relación entre el orden parcial $\leq$ sobre $\mathcal{T}$ y el orden parcial $\preceq$ sobre $\mathcal{T}^*$ hace que la jerarquía $\preceq$ sea nula si lo es $\leq$. En este caso, además, una regla con tipo cartesiano $t_1 \otimes t_2$ será aplicable sobre dos argumentos con tipo cartesiano asociado $x_1 \otimes x_2$ si y sólo si $x_1 = t_1, x_2 = t_2$, ya que no hay ninguna relación de orden entre los miembros de $\mathcal{T}$.

Así pues, el funcionamiento de una regla genérica $\mathcal{G}$ definida sobre un dominio lingüístico $\mathcal{T}$ no jerarquizado es enteramente equivalente a una gramática libre de contexto anotada. Cada regla parcial equivale a una regla libre de contexto con una función de combinación de información asociada.

En el caso de una regla genérica para la especificación de tipos sintácticos como las que se analizan mediante el algoritmo 6.8, ésta es equivalente a una gramática libre de contexto en forma normal de Chomsky. Los tipos de $\mathcal{T}$ son los símbolos de esa gramática, y cada regla parcial equivale a una regla libre de contexto cuya parte izquierda es el resultado de aplicar la regla parcial, y cuya parte derecha son las condiciones de tipo sobre los argumentos.

Es decir, la regla genérica:

$$\mathcal{SYN} = \begin{cases} SYN_{\mathsf{A}_1 \otimes \mathsf{B}_1}(\mathsf{X},\mathsf{Y}) = \mathsf{C}_1 \\ SYN_{\mathsf{A}_2 \otimes \mathsf{B}_2}(\mathsf{X},\mathsf{Y}) = \mathsf{C}_2 \\ \cdots \\ SYN_{\mathsf{A}_n \otimes \mathsf{B}_n}(\mathsf{X},\mathsf{Y}) = \mathsf{C}_n \end{cases} \tag{6.12}$$

es isomorfa a la gramática libre de contexto en forma normal de Chomsky :

$$\begin{cases} C_1 \rightarrow A_1 B_1 \\ C_2 \rightarrow A_2 B_2 \\ \cdots \\ C_3 \rightarrow A_3 B_3 \end{cases} \tag{6.13}$$

El proceso de parsing de abajo arriba también es equivalente para ambas gramáticas. Las reglas **reduce** que habíamos especificado para reconocer gramáticas libres de contexto y reglas genéricas son equivalentes en el límite de jerarquía nula. La regla de reducción para reglas genéricas

$$\frac{[\alpha \mathsf{x}_1 \mathsf{x}_2 \bullet, j]}{[\alpha \mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2) \bullet, j]} \mathbf{SYN}(\mathsf{x}_1,\mathsf{x}_2)(\mathsf{x}_1,\mathsf{x}_2) \neq \perp \tag{6.14}$$

se vuelve idéntica a:

$$\frac{[\alpha BC \bullet, j]}{[\alpha A \bullet, j]} \quad A \rightarrow BC \tag{6.15}$$

Naturalmente, esta identidad no es posible sólo para el algoritmo en particular de shift-reduce, sino que es posible para cualquier algoritmo de abajo arriba que reconozca gramáticas libres de contexto. Por lo tanto, la complejidad del proceso de parsing en el límite de jerarquía nula, es la misma que la del parsing de una CFG, es decir, n^3.

A continuación vamos a especificar un método de compilación de reglas genéricas en gramáticas sintagmáticas que se basa en la equivalencia entre estas dos cuando se produce una desestructuración de los dos conjuntos $\mathcal{T}$ y $\mathcal{T}^*$.

6.2.2 Algoritmo de reescritura

Consideremos el poset $\mathcal{T}$ de la figura 6.1 y un poset cartesiano asociado $\mathcal{T}^*$. La notación $x_1 \otimes x_2 : f$ quiere decir que el tipo $x_1 \otimes x_2$ está asociado a la función f.

Figura 6.1: $\mathcal{T}$ y $\mathcal{T}^*$

Para realizar la compilación, comenzaremos por las ramas más bajas de $\mathcal{T}^*$; en particular, por los nodos de los que sólo cuelgan terminales. En el caso de $\mathcal{T}^*$ se trata del nodo np⊗vp, que tiene una única hoja $\text{np}\otimes\text{vp}_1$. Para que no se produzcan conflictos, hay que cambiar la especificación de tipos np⊗vp asociada a la función g para que g sólo sea aplicable en aquellos lugares donde no entre en conflicto con h, asociada al tipo $\text{np}\otimes\text{vp}_1$. En este caso, es obvio que los casos complementarios a $\text{np}\otimes\text{vp}_1$ dentro de los que abarca np⊗vp se reducen al tipo $\text{np}\otimes\text{vp}_2$. Podemos reescribir $\mathcal{T}^*$ según se muestra en la figura 6.2

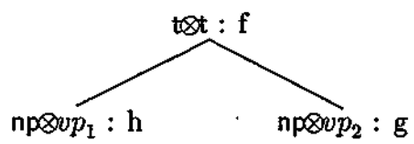


Figura 6.2: Primer paso en la reescritura de $\mathcal{T}^*$

El comportamiento del nuevo poset cartesiano es idéntico al del antiguo. Pero hemos eliminado el comportamiento no monótono de la regla genérica asociada en los casos en los que tanto g como h eran aplicables, y se escogía esta última por ser más específica.

Aplicando de nuevo el mismo procedimiento sobre el tipo cartesiano t⊗t al nuevo poset cartesiano, obtenemos el poset cartesiano de la figura 6.3. En este caso, ya no hay ningún ninguna jerarquización entre sus tipos. Por lo tanto, cuando la regla genérica sea invocada existirá, como mucho, una regla parcial

aplicable. No hay, pues, necesidad de establecer precedencia entre reglas al invocar la gramática.

---

np⊗vp_1 : h np⊗vp_2 : g vp⊗t : f np⊗np: f

Figura 6.3: Segundo y último paso en la reescritura de $\mathcal{T}^*$. Este poset cartesiano es totalmente equivalente a $\mathcal{T}^*$ como función genérica, pero no hay ninguna dependencia jerárquica entre reglas. En cada caso, habrá a lo sumo una regla aplicable.

Nuestro algoritmo de conversión consiste en aplicar sucesivamente este proceso de "aplanamiento" sobre el último nivel de ramificación del poset cartesiano, hasta convertirlo en un conjunto sin jerarquización.

Llamemos $S(a)$ al conjunto de todas las subclases de un tipo dado a. Llamemos *altura* de un tipo a a la distancia máxima de a a cualquiera de sus subtipos terminales (las hojas de una jerarquía tienen, según esta convención, jerarquía cero).

El paso fundamental de aplanamiento consiste en determinar los tipos equivalentes a cada tipo a de altura 1 que complementen a los subtipos de a:

$$\mathcal{COMP}(a) \equiv \bigcup_{s \leq a} comp(s, a) \tag{6.16}$$

donde $comp(s, a)$ es el conjunto de los tipos que complementan a s según a. Si $a =$ l⊗r, $s =$ x⊗y , el conjunto $comp(s, a)$ se divide en dos: aquellos de la forma x⊗tipo y aquellos de la forma tipo⊗y. En la regla genérica, cada nuevo tipo $t \in \mathcal{COMP}(a)$ lleva asociado la misma función que a. Por construcción, tienen los mismos supertipos que a, pero ya no son supertipos de los subtipos directos de a.

Las siguientes definiciones no hacen más que especificar cuáles pueden ser los tipos cartesianos que representa la variable tipo en las dos expresiones anteriores.

$$comp(\text{x⊗y}, \text{l⊗r}) \equiv comp^{right}(\text{x⊗y}, \text{l⊗r}) \cup comp^{left}(\text{x⊗y}, \text{l⊗r}) \tag{6.17}$$

$$comp^{right}(\mathsf{x}\otimes\mathsf{y},\mathsf{l}\otimes\mathsf{r}) \equiv \bigcup_{y\leq t\leq r} N_y^{right}(x,t) \tag{6.18}$$

$$comp^{left}(\mathsf{x}\otimes\mathsf{y},\mathsf{l}\otimes\mathsf{r}) \equiv \bigcup_{x\leq t\leq l} N_x^{left}(t,y) \tag{6.19}$$

$$N_y^{right}(x,t) \equiv \{\mathsf{d}\otimes\mathsf{y} \mid x \not\leq d \leq t\} \tag{6.20}$$

$$N_x^{left}(t,y) \equiv \{\mathsf{x}\otimes\mathsf{d} \mid y \not\leq d \leq t\} \tag{6.21}$$

Veamos qué valores toman estas expresiones en un ejemplo. Consideremos los tipos cartesianos $\mathsf{x}_1\otimes\mathsf{x}_2 \preceq \mathsf{y}_1\otimes\mathsf{y}_2$ sobre la jerarquía $\langle\leq, T\rangle$ de la figura 6.4.

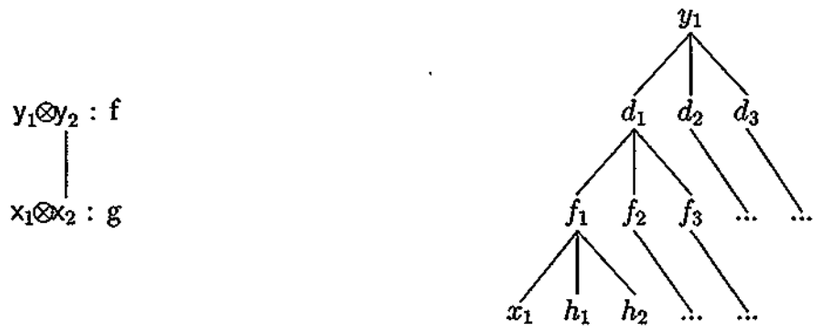


Figura 6.4: Jerarquía asociada a los tipos cartesianos $\mathsf{x}_1\otimes\mathsf{x}_2, \mathsf{y}_1\otimes\mathsf{y}_2$

Los términos que intervienen en la formación de $comp^{right}(\mathsf{x}_1\otimes\mathsf{x}_2, \mathsf{y}_1\otimes\mathsf{y}_2)$ (y con ellos queda ilustrado también el funcionamiento de $comp^{left}(\mathsf{x}_1\otimes\mathsf{x}_2, \mathsf{y}_1\otimes\mathsf{y}_2)$, que es la versión simétrica del primero) son los siguientes:

$$\begin{aligned} comp^{right}(\mathsf{x}_1\otimes\mathsf{x}_2, \mathsf{y}_1\otimes\mathsf{y}_2) &= N_{\mathsf{x}_2}^{right}(\mathsf{x}_1,\mathsf{d}_1) \cup N_{\mathsf{x}_2}^{right}(\mathsf{x}_1,\mathsf{f}_1) \\ N_{\mathsf{x}_2}^{right}(\mathsf{x}_1,\mathsf{d}_1) &= \{\mathsf{f}_2\otimes\mathsf{x}_2 : f, \mathsf{f}_3\otimes\mathsf{x}_2 : f\} \\ N_{\mathsf{x}_2}^{right}(\mathsf{x}_1,\mathsf{f}_1) &= \{\mathsf{h}_1\otimes\mathsf{x}_2 : f, \mathsf{h}_2\otimes\mathsf{x}_2 : f\} \end{aligned} \tag{6.22}$$

Nótese que todos los tipos de $\mathcal{COMP}(\mathsf{y}_1\otimes\mathsf{y}_2)$ llevan asociada la misma función f que llevaba asociada $\mathsf{y}_1\otimes\mathsf{y}_2$.

A partir de estas definiciones, podemos describir el proceso de reducción de un poset cartesiano $\mathcal{T}^*$ por el cual descomponemos el nivel inferior de jerarquización:

$$\mathcal{REDUCCION}(\mathcal{T}^*) \equiv R(\mathcal{T}^*) \cup \bigcup_{\substack{a \in \mathcal{T}^* \\ altura(a) = 1}} \mathcal{COMP}(a) \tag{6.23}$$

donde

$$R(\mathcal{T}^*) \equiv \{t \in \mathcal{T}^* | altura(t) \neq 1\} \tag{6.24}$$

Ahora estamos en condiciones de especificar el algoritmo de conversión, por medio de una definición recursiva:

Algoritmo de conversión

$$\mathcal{CONVERSION}(\mathcal{T}^*) \equiv \begin{cases} \mathcal{T}^* \text{ si } \forall t \in \mathcal{T}^* altura(t) = 0 \\ \mathcal{CONVERSION}(\mathcal{REDUCCION}(\mathcal{T}^*)) \\ \text{en caso contrario.} \end{cases} \tag{6.25}$$

La función $\mathcal{CONVERSION}$ realiza la compilación de la regla genérica asociada a $\mathcal{T}^*$ en otra equivalente cuyo comportamiento es monótono.

Puede verse que el algoritmo no se ve afectado por el hecho de que haya herencia múltiple en el poset cartesiano $\mathcal{T}^*$ ni en la jerarquía original $\mathcal{T}$. En $\mathcal{T}^*$, las dependencias múltiples van desapareciendo gradualmente igual que el resto. Las de la jerarquía original, obviamente, no constituyen un problema.

6.2.3 Revisión para universos abiertos

Hay que hacer, sin embargo, una acotación al funcionamiento del sistema según la interpretación que se haga de la jerarquía léxica. El algoritmo funciona correctamente si $\mathcal{T}$ se interpreta bajo una *closed world assumption* o *hipótesis de*

universo cerrado. En una jerarquía de universo cerrado, cada objeto pertenece exactamente a un tipo mínimo. Esto implica que cada objeto que pertenece a la denotación de un tipo no mínimo es descrito, al menos, por uno de sus subtipos. Si, por el contrario, se adopta una filosofía de universo abierto, en la que puede haber objetos que pertenezcan directamente a un tipo no mínimo sin pertenecer a la vez a alguno de sus subtipos, el algoritmo no construye adecuadamente el conjunto de tipos complementarios a uno dado.

Consideremos el caso de la figura 6.5.

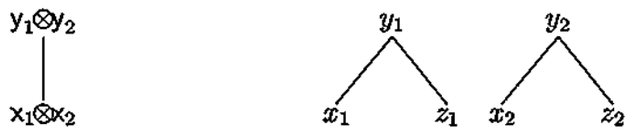


Figura 6.5: El conjunto complementario de $y_1 \otimes y_2$ con respecto a $x_1 \otimes x_2$ no es correcto si se interpreta la jerarquía $\mathcal{O}$ como un universo abierto.

Puede comprobarse que, para esta jerarquía,

$$\mathcal{COMP}(y_1 \otimes y_2) = \{x_1 \otimes z_2, z_1 \otimes x_2\} \tag{6.26}$$

Si se interpreta $\mathcal{O}$ como un universo abierto, esta especificación no es correcta, pues no considera los casos en los que uno de los dos argumentos pertenece directamente a y_1 o y_2.

Para tratar el caso de los universos abiertos, es necesario reescribir de antemano la jerarquía léxica $\mathcal{T}$ en un sistema cerrado equivalente. Para ello, basta con introducir un nuevo tipo mínimo $t_n \leq n$ por cada elemento no terminal n de la jerarquía, y asignar todos los objetos que pertenecen directamente a n como pertenecientes al tipo mínimo t_n.

En la figura 6.6 se ilustra la reescritura del poset $\mathcal{O}$ de 6.5.

En este caso, el conjunto $\mathcal{COMP}(y_1 \otimes y_2)$ es diferente:

$$\mathcal{COMP}(y_1 \otimes y_2) = \{x_1 \otimes z_2, z_1 \otimes x_2, y_{1n} \otimes x_2, x_1 \otimes z_{2n}\} \tag{6.27}$$

Figura 6.6: La interpretación de esta jerarquía como universo cerrado es equivalente a la de $\mathcal{O}$ como universo abierto

6.3 Recapitulación

Los sistemas de parsing de arriba abajo para gramáticas libres de contexto son aplicables al parsing de reglas genéricas. El coste sigue siendo el mismo en el tamaño de la cadena, pero aparece una dependencia cuadrática (en el peor de los casos) con el tamaño de la jerarquía léxica. Esta dependencia no surge debido al comportamiento no monótono de las reglas genéricas, sino a la jerarquización del dominio de objetos lingüísticos; cualquier gramática monótona que opere sobre un dominio jerarquizado tiene ese factor de complejidad. La dependencia con el tamaño de la gramática (es decir, con el número de reglas parciales asociadas a una regla genérica) es lineal. Cuando se utiliza una gramática libre de contexto para dar estructura a las oraciones y una regla genérica para realizar la composición semántica, aparece un factor lineal en el tamaño de la gramática libre de contexto y otro en el número de reglas parciales asociadas a la regla genérica.

Estos razonamientos no consideran la complejidad asociada a la aplicación de las reglas parciales individuales, cuya naturaleza apenas hemos restringido. La complejidad del parsing es la que hemos mencionado cuando las reglas genéricas se aplican en tiempo constante, como es el caso de una regla genérica que simplemente combina categorías sintácticas.

Parte III

Algunas aplicaciones

Capítulo 7

Reglas genéricas y coordinación de no constituyentes

En este capítulo presentamos una regla genérica para dar cuenta de los fenómenos sintácticos asociados con la coordinación de no constituyentes dentro del entorno de las gramaticas categoriales.

Las gramaticas categoriales constituyen un entorno lingüístico muy apropiado para su reformulación mediante reglas genéricas. En él, los objetos lingüísticos básicos contienen la información sobre su comportamiento sintáctico, y existe sólo una regla básica que relaciona esa información: la aplicación funcional. Mediante esta regla se dan cuenta de muchos fenómenos sintácticos, y la complejidad computacional del sistema (así como su capacidad generativa) son equivalentes a los de las gramáticas libres de contexto. Para dar cuenta de fenómenos complejos, como las dependencias a larga distancia o la coordinación de no constituyentes, se postulan nuevas reglas y operadores de gran potencia, que vuelven intratables los procesos de parsing asociados a esas gramáticas y, con frecuencia, presentan problemas graves de sobregeneración. Estos problemas se suelen paliar mediante mecanismos extragramaticales (como heurísticas ad-hoc introducidas en el algoritmo de parsing). El problema de representación es que no hay ninguna forma de expresar directamente en la gramática el carácter regular o excepcional de cada regla, ni los fenómenos a los que se circunscribe su aplicación. La aplicabilidad de las gramáticas categoriales al procesamiento del lenguaje natural queda mermada, de esta forma, por las dificultades de procesamiento.

El formalismo de reglas genéricas proporciona una solución a este problema, permitiendo la introducción de cada regla a su nivel adecuado de especificidad, y estableciendo preferencias de aplicación según esta especificidad. Este conocimiento se introduce en el nivel de descripción gramatical de forma declarativa.

En este capítulo reformulamos una gramática categorial para la coordinación de no constituyentes basada en el operador *tupla* [Solias, 1992]. Mientras que la

formulación original es computacionalmente intratable, la regla genérica asociada puede ser procesada con las mismas técnicas que una gramática libre de contexto. El conocimiento necesario para mejorar de esta forma los procesos de parsing está lingüísticamente motivado y se expresa en el nivel de representación gramatical.

7.1 Coordinación de no constituyentes

El esquema general para la coordinación corresponde a la conjunción de constituyentes del mismo tipo:

Nada es seguro, excepto la muerte *y* los impuestos.

La seguiría por tierra *y* por mar.

Toma el dinero *y* corre.

El largo *y* cálido *verano.*

Sin embargo, hay casos en los que los elementos coordinados no son sintagmas, y requieren un tratamiento específico:

(a) Eva soñó con la nieve en Santa Cruz *y* con el mar en Madrid.

(b) Perdí el paraguas el jueves *y* el chubasquero el viernes.

(c) Ella aborrece *y* yo disfruto *los días lluviosos.*

"El paraguas el jueves" o "con el mar en Madrid" no se consideran sintagmas aisladamente. Sin embargo, pueden ser miembros de una coordinación, como en los ejemplos anteriores.

La mayoría de las soluciones que se han propuesto para resolver este tipo de fenómenos postulan, por encima de las reglas de la gramática, un segundo nivel de representación "metagramatical" en el que metarreglas, algoritmos heurísticos, transformaciones, etc, dan cuenta de los fenómenos de coordinación. El primer sistema que daba cuenta de un rango amplio de fenómenos relacionados con la coordinación fue SYSCONJ [Woods-73, 1973], en el que se alcanzaba una cobertura razonable de los fenómenos en cuestión mediante ATN adaptadas para incluir mecanismos procedurales de parsing. En [Dahl and McCord, 1983], las reglas para tratar la coordinación se encuentran embebidas en el algoritmo de parsing. [Fong, 1985] proponen una solución general para casi todos los fenómenos relacionados con la coordinación mediante un mecanismo extragramatical de detección de "sintagmas equivalentes". En [Lavelli and Stock, 1990] se solucionan

simultáneamente los problemas de coordinación y elipsis mediante una reformulación de los algoritmos de parsing mediante tablas (*chart parsing*), introduciendo una nueva función de selección. En [Milward, 1990] se postulan dos niveles de representación gramatical, en los que axiomas y reglas combinan una categoría con una cadena de palabras para formar otra categoría. Se permite la coordinación, no de categorías idénticas, sino de cadenas que puedan producir las mismas transiciones entre categorías. En [Milward, 1994] se introduce el conocimiento necesario para procesar coordinaciones al nivel del parsing, pero representando este nivel declarativamente mediante las *Dynamic Grammars*, que estudian estados de parsing y transiciones entre estados. En [Dahl et al., 1995] se utiliza un mecanismo metagramatical de identificación de estructuras paralelas en el nivel sintáctico y en el nivel semántico. En este trabajo, los criterios de paralelismo no sólo reducen el espacio de búsqueda, sino que contribuyen activamente a encontrar una solución.

La otra aproximación básica a la coordinación de no constituyentes pasa por relajar el concepto de constituencia. Éste es el caso de las propuestas hechas en el entorno de las gramáticas categoriales, que vamos a estudiar a continuación. Estas propuestas dan cuenta de los fenómenos de coordinación sin postular nuevos niveles de representación o nuevos mecanismos de parsing. Sin embargo, su coste computacional es muy elevado y producen ambigüedades espurias.

Nuestra propuesta reformula una gramática categorial que trata el fenómeno de la coordinación de no constituyentes. La regla genérica que resulta tiene las ventajas de las aproximaciones algorítmicas y de las aproximaciones que relajan el concepto de constituencia: a) permite establecer el conocimiento sobre la coordinación al nivel gramatical, y b) no produce ambigüedades espurias y puede ser procesada eficientemente.

7.2 Coordinación de no constituyentes y gramáticas categoriales

Aquí vamos a exponer las soluciones que se han propuesto dentro de la teoría lingüística de las gramáticas categoriales. Para ello recordaremos primero los principios básicos de esta teoría.

7.2.1 El cálculo Lambek

En [Lambek, 1958] se propusieron los fundamentos algebraicos para el formalismo categorial desarrollado por [Ajdukiewicz, 1935] y [Bar-Hillel, 1953]. En este sistema, los tipos lingüísticos se generan recursivamente a partir de unos pocos tipos atómicos (N para nombres comunes, NP para predicados nominales, S para

oraciones ...) por medio de los operadores binarios '/' y '\' Los tipos complejos formados por estos operadores tienen la siguiente interpretación:

$$\begin{aligned} D(C \,/\, B) &= \{x_1 : \forall x_2 \in D(B), x_1 + x_2 \in D(C)\} \\ D(A\backslash C) &= \{x_1 : \forall x_2 \in D(A), x_2 + x_1 \in D(C)\} \end{aligned} \tag{7.1}$$

Los operadores '/'y '\'nos permiten asignar categorias para determinantes np/n, verbos intransitivos '$np\backslash s$', verbos transitivos '$(np\backslash s)/np$', modificadores de predicados verbales '$(np\backslash s)\backslash(np\backslash s)$', etc.

En 7.2 se muestra el cálculo de secuentes para esos tipos, que es completo respecto a la semántica de 7.1 y 7.2. Un secuente de la forma $A_1, ..., A_n \Rightarrow B$ significa que la concatenación ordenada de todas las cadenas en $A_1, ..., A_n$ produce una cadena en B. Un ejemplo de un secuente lingüísticamente relevante es una secuencia de tipos que forme una oración. Por ejemplo, $np, (np\backslash s)/np, np \Rightarrow s$ significa que la concatenación de categorías $np, (np\backslash s)/np, np$ forma una oración. Hay una regla de uso (L) y una regla de prueba (R) asociada a cada operador. En esta presentación, Γ, Δ representan secuencias (quizás vacías) de tipos. T representa secuencias no vacías. Los símbolos A, B, C representan tipos no vacíos:

$$\frac{}{A \Rightarrow A} Ax$$

$$\frac{\Gamma, B \Rightarrow C}{\Gamma \Rightarrow C/B} R/ \qquad \frac{T \Rightarrow B \quad \Gamma, C, \Delta \Rightarrow D}{\Gamma, C/B, T, \Delta \Rightarrow D} L/ \tag{7.2}$$

$$\frac{A, \Gamma \Rightarrow C}{\Gamma \Rightarrow A\backslash C} R\backslash \qquad \frac{T \Rightarrow A \quad \Gamma, C, \Delta \Rightarrow D}{\Gamma, T, A\backslash C, \Delta \Rightarrow D} L\backslash$$

Las reglas siguientes son teoremas fácilmente derivables del cálculo en 7.2:

Aplicación hacia adelante: $X/Y \;\; Y \Rightarrow X$
Aplicación hacia atrás: $Y \;\; Y\backslash X \Rightarrow X$
Composición hacia adelante: $X/Y \;\; Y/Z \Rightarrow X/Z$
Composición hacia atrás: $Z\backslash Y \;\; Y\backslash X \Rightarrow Z\backslash X$
Elevación de tipo hacia adelante: $X \Rightarrow Y/(X\backslash Y)$
Elevación de tipo hacia atrás: $X \Rightarrow (Y/X)\backslash Y$ (7.3)

He aquí algunas de las derivaciones para esos teoremas:

Composición hacia adelante

```
          ----- Ax  ------ Ax
          Y -> Y    X -> X
  ------Ax ---------------- L/
  Z -> Z   X/Y  Y -> X
  ------------------------- L/
  X/Y  Y/Z  Z -> X
  ------------------------- R/
  X/Y  Y/Z -> X/Z
```

Composición hacia atrás

```
  ----Ax   ------ Ax
  X -> X   Y -> Y
  -------------- L\
  X   X \ Y -> Y
  -------------- R/
  X -> Y/(X\Y)
```

7.2.2 Elevación de tipos y composición

En muchas aplicaciones lingüísticas de la gramática categorial, estos teoremas que acabamos de mostrar se usan como reglas de la gramática. Las dos reglas básicas son las de aplicación funcional:

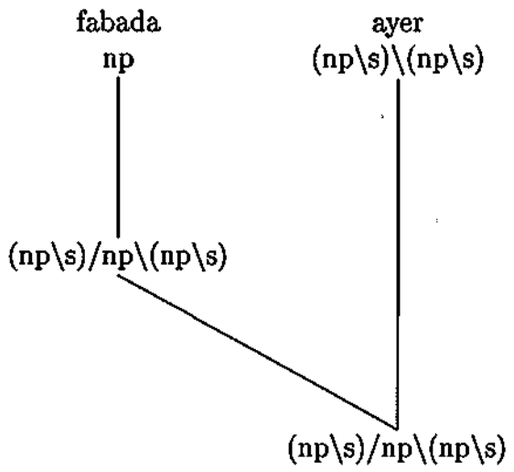
X/Y Y → X (aplicación hacia adelante)

Y Y\X → X (aplicación hacia atrás)

Estas dos reglas son las únicas que no hacen uso de la propiedad asociativa del cálculo Lambek; una gramática categorial con estas dos reglas es equivalente en poder generativo a una gramática libre de contexto. La coordinación de constituyentes, por ejemplo, puede hacerse en gramáticas categoriales mediante estas dos reglas, asignando a la conjunción el esquema de categoría '(X \ X) / X'.

La propiedad asociativa en la estructura de una oración implica el desvanecimiento del concepto de árbol de análisis. La incorporación de reglas que se derivan haciendo uso de esta propiedad permite dar cuenta, de forma elegante, de fenómenos más complejos que el uso exclusivo de las reglas de aplicación, pero produce ambigüedades espurias y fenómenos de sobregeneración.

En [Dowty, 1988] se utilizan este tipo de reglas para dar cuenta de la coordinación de no constituyentes. En ese trabajo, Dowty propone el uso de las dos reglas de aplicación funcional mientras sea posible. Cuando una oración no pueda ser analizada usando exclusivamente la aplicación funcional, se introducirán la composición y la elevación de tipos en el conjunto de reglas utilizables. La razón para no disponer de esas reglas desde un primer momento es que son excesivamente poderosas. La regla de elevación de tipos, por ejemplo, puede aplicarse sobre cualquier categoría para producir diferentes categorías. La regla de composición, actuando en conjunción con la regla de elevación de tipos, permite formar constituyentes con cualquier par de tipos, combinando categorías que nunca podrían hacerlo por medio de las reglas de aplicación funcional. Por ejemplo, es posible combinar un *np* y un modificador de predicado verbal concatenados elevando el tipo de *np* y aplicando composición funcional sobre el tipo elevado y el modificador de predicado verbal:



(7.4)

En 7.4, elevamos el tipo de *np* sustituyendo $np\backslash s$ por la variable Y del esquema general de elevación de tipos. Así, obtenemos $(np\backslash s)/np\backslash(np\backslash s)$, y podemos componer este tipo con $(np\backslash s)\backslash(np\backslash s)$, obteniendo $(np\backslash s)/np\backslash(np\backslash s)$, que cancelará un verbo a su izquierda para formar un predicado verbal (es decir, $np\backslash s$) [1]

[1] En adelante usaremos la abreviatura *vp* para el tipo $np\backslash s$, y la abreviatura *vm* para $vp\backslash vp$,

En el caso regular, Dowty trata la coordinación usando exclusivamente las reglas de aplicación. Pero en casos como

Juan comió fabada ayer y pisto hoy. (7.5)

Dowty propone el uso de la regla de elevación de tipo y la regla de composición. De esta forma, Dowty da cuenta de la coordinación de no constituyentes de una forma sencilla y elegante. Eleva y compone las categorias situadas a ambos lados de la conjunción y, a continuación, coordina los tipos equivalentes que resultan a uno y otro lado. En la figura 7.1 se aprecia este proceso para la oración 7.5 [2].

En el análisis de la figura 7.1, hemos elevado la categoría de cada *np* mediante la sustitución descrita en 7.4, y las hemos compuesto con su correspondiente modificador de predicado adverbial. De esta manera, conseguimos la combinación de los tipos, aparentemente incombinables, np, $vp\backslash vp$. A continuación, cancelamos la categoría de la conjunción, obteniendo un único tipo $((np\backslash s)/np)\backslash vp$ que cancela al verbo transitivo dando un *vp*.

El uso de la elevación de tipos y la composición nos permite dar cuenta de los fenómenos de coordinación de no constituyentes de una forma muy elegante. Sin embargo, se hace a costa de introducir dos reglas que vuelven el proceso de parsing intratable. La regla de elevación de tipos, además de poder aplicarse sobre cualquier categoría X en cualquier momento del proceso de análisis, deja indeterminada la categoría Y del tipo sobre el que se eleva X. Si se contempla el proceso desde la perspectiva de los cálculos de secuentes, existen razones independientes para no usar la secuencia de reglas de elevación de tipo y composición funcional (cf. [Moortgat, 1988, Solias, 1992, Morrill, 1994]). A continuación mostramos una forma más sencilla de tratar los fenómenos de coordinación de no constituyentes dentro de las gramáticas categoriales.

7.2.3 El producto tupla

En [Solias, 1993, Morrill and Solias, 1993] se introduce el producto tupla como una forma de tratar los fenómenos de vaciado asociados a la coordinación. Ese mismo operador puede usarse para tratar fácilmente la coordinación de no constituyentes [Gonzalo and Solias, 1995]. Recordemos los ejemplos relevantes:

(a) Eva soñó *con la nieve en Santa Cruz* y *con el mar en Madrid.*

(b) Perdí *el paraguas el jueves* y *el chubasquero el viernes.*

siempre que sea necesario.

[2] En la derivación 7.5, la abreviatura CONJ resume la cancelación del esquema de tipo $(x\backslash x)/x$ mediante aplicación hacia adelante y aplicación hacia atrás.

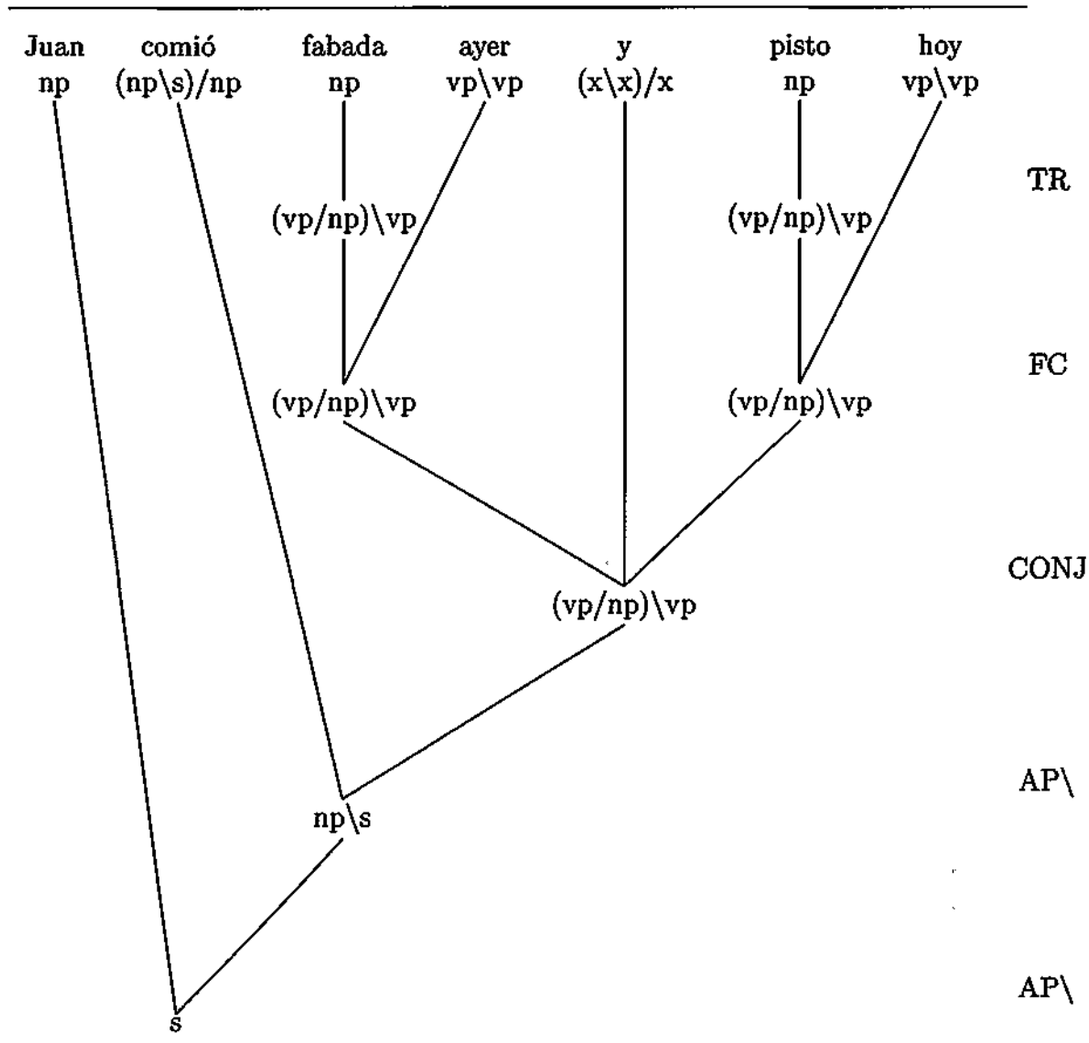


Figura 7.1: Análisis de 'Juan comió fabada ayer y pisto hoy' mediante las reglas de elevación de tipos y composición funcional.

(c) *Ella aborrece* y *yo adoro* los días lluviosos.

En estas oraciones, podemos considerar *'con la nieve en Santa Cruz'*, *'con el mar en Madrid'*, *'el paraguas el jueves'*, *'ella aborrece'*, etc, como tuplas de expresiones, pertenecientes a un operador tupla. En este caso, el esquema de tipo de la conjunción, $(x\backslash x)/x$ sustituiría la variable x por un producto tupla.

Para introducir este operador es necesario extender el álgebra básica de tipos, añadiendo una operación tupla. Así, el álgebra será ahora $(L^*, +, \langle ., . \rangle)$, donde $+$ es la operación de concatenación y $\langle ., . \rangle$ la operación de formación de tuplas.

La definición para el producto tupla es como sigue:

$$D(A \diamond B) = \{\langle x_1, x_2 \rangle : x_1 \in D(A), x_2 \in D(B)\} \tag{7.6}$$

Las reglas del cálculo de secuentes correspondientes a 7.6 son:

$$\frac{\Gamma \Rightarrow A \quad \Delta \Rightarrow B}{\Gamma, \Delta \Rightarrow A \diamond B} R\diamond \qquad \frac{\Gamma, A, B, \Delta \Rightarrow C}{\Gamma, A \diamond B, \Delta \Rightarrow C} L\diamond \tag{7.7}$$

La derivación del ejemplo más sencillo que involucra la coordinación de no constituyentes sería como se muestra en la figura 7.2.

Los ejemplos con más de dos elementos en la tupla necesitan una generalización para considerar n-tuplas. Esa generalización es inmediata, usando la definición recursiva usual de n-tupla: $\langle x_1, ..., x_n \rangle = \langle \langle x_1, ..., x_{n-1} \rangle x_n \rangle$.

Para casos como (c) (a los que se les llama *elevación del nodo derecho*, el análisis procedería de forma similar. En primer lugar, la conjunción cancelaría una tupla formada por un *np* y un verbo, y el tipo resultante se combinaría con los complementos del predicado verbal.

7.3 Una regla genérica para analizar la coordinación de no constituyentes

Como hemos visto, Dowty introduce dos nuevas reglas cuyo coste computacional es muy alto. Su indicación informal para licenciar la aplicación de estas reglas sólo cuando un primer proceso de reconocimiento, basado exclusivamente en las reglas de aplicación funcional, no consigue analizar la oración, no es suficiente para

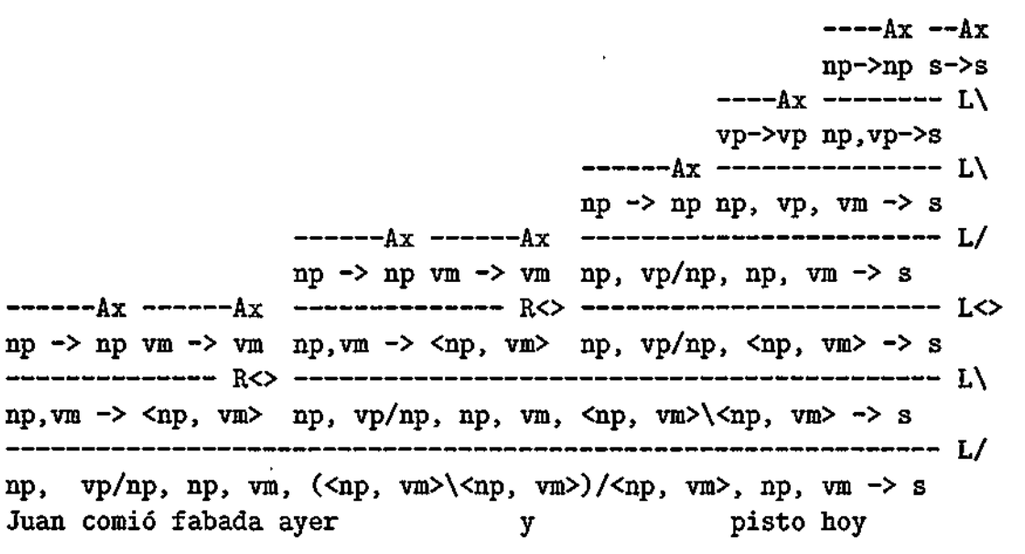


Figura 7.2: Derivación de "Juan comió fabada ayer y pisto hoy"por medio de los operadores tupla. *vp* es una abreviación para $np\backslash s$, y *vm* es una abreviación para $vp\backslash vp$.

restringir su aplicación cuando hay que analizar una coordinación de no constituyentes. Algunos de los problemas que se presentan con el uso de la elevación de tipos y la composición funcional pueden resolverse mediante el operador tupla y las reglas asociadas a este operador en un cálculo de secuentes. Sin embargo, la regla de introducción del operador tupla puede ser aplicada sobre cualquier par de categorías. Este hecho mantiene los procesos de parsing intratables, y no refleja las condiciones, muy especiales, en las que la tupla debe ser introducida.

El concepto de regla genérica ofrece una forma natural de expresar las condiciones y reglas asociadas al parsing de oraciones con posibles coordinaciones de no constituyentes. La estructuración de las reglas sintácticas como un conjunto de reglas por defecto es especialmente adecuada en un entorno en el que la mayor parte del trabajo de asociación sintáctica se hace mediante una única regla de aplicación funcional, y reglas distintas como las relacionadas con el operador tupla se usan solo en ocasiones específicas.

Las dos reglas esenciales que queremos expresar son:

- *Por defecto, dos tipos se combinan mediante aplicación funcional. Si la aplicación funcional no es aplicable, no pueden ser combinados.*
- *Cuando intentamos combinar dos complementos verbales y hay una conjunción precediéndolos, debe formarse una tupla con esos complementos.*

"Complemento verbal", para nuestros propósitos actuales, se reduce a los tipos np, vm o, recursivamente, al tipo tupla.

Esta clase de reglas garantizarían que una tupla se forma sólo en los casos en los que es necesaria para el proceso de análisis. Sólo necesitamos una regla adicional para identificar de forma óptima los elementos correspondientes a la tupla a la izquierda de la conjunción.

El formalismo de reglas genéricas permite expresar directamente tales especificaciones. Una vez expresadas como una regla genérica,pueden reconocerse con las mismas técnicas que una gramática libre de contexto.

El primer paso para reformular el tratamiento de la coordinación de no constituyentes es expresar las operaciones relacionadas con la gramática en forma de reglas binarias, para poder adaptarlas al formato de las reglas parciales:

$$\begin{aligned}
&\textbf{fa} : X/Y \quad Y \rightarrow \quad X \\
&\textbf{ba} : Y \quad Y\backslash X \quad \rightarrow \quad X \\
&\mathbf{I}_{\langle ... \rangle} : conj \quad X \quad Y \rightarrow \quad conj \quad \langle X, Y \rangle \\
&\textbf{scan} : X_n \quad \langle \langle X_1 \cdots X_{n-1} \rangle X_n \rangle \backslash \langle X_1 \cdots X_m \rangle \rightarrow \langle X_1 \cdots X_{n-1} \rangle \backslash \langle X_1 \cdots X_m \rangle \\
&\mathbf{D}_{\langle ... \rangle} : \langle X_1 \cdots X_n \rangle \rightarrow X_1 \cdots X_n
\end{aligned} \tag{7.8}$$

fa y **ba** son las reglas usuales de aplicación hacia adelante y hacia atrás. $\mathbf{I}_{\langle ... \rangle}$ es la regla para introducir una tupla. Así formulada, podría parecer una regla sensible al contexto, ya que la formación de una tupla sólo es posible cuando una conjunción está presente a la izquierda de los elementos que van a formar la tupla. Sin embargo, esta sensibilidad al contexto no introduce ninguna complejidad adicional al reconocimiento libre de contexto, ya que el elemento contextual que licencia la regla es un elemento léxico, cuya presencia puede ser comprobada en un tiempo constante. $\mathbf{I}_{\langle ... \rangle}$ implementa la secuencia de reglas $L/$ y $R\diamond$ en $(\langle A, B\rangle\backslash\langle A, B\rangle)/\langle A, B\rangle$ A, B, mostrado en (7.2). **scan** encuentra los elementos paralelos a la tupla en la parte izquierda de la conjunción. Esta reformulación de la operación de cancelación de tupla mantiene la formulación binaria que estamos buscando y tiene en cuenta, por otra parte, la asimetría entre los procesos de formación para los conjuntos coordinados a izquierda y derecha de la conjunción [3]. La necesidad de construir una tupla viene dada por los elementos que aparecen en el conjunto coordinante de la derecha, mientras que el conjunto coordinante de la izquierda se construye para corresponder al otro. **scan** introduce la secuencia de reglas $L\backslash$ y $R\diamond$ en 7.2. $\mathbf{D}_{\langle ... \rangle}$ es la regla de eliminación del operador tupla, y hace la función de la regla $L\diamond$ del cálculo de secuentes.

Un punto esencial para escribir una regla genérica basada en las reglas de 7.8 es determinar los tipos de los argumentos asociados a cada operación de combinación.

Por defecto, cualquier combinación de categorías se basa en las reglas de aplicación funcional: aplicación hacia adelante, **fa** , y aplicación hacia atrás, **ba** . Por lo tanto, la primera regla parcial será aplicable sobre un par de argumentos del tipo más general, T. Esta regla tratará de aplicar **fa** o **ba** , y devolverá $\perp$ si falla (prohibiendo esa combinación).

La segunda regla parcial que debemos considerar es la de formación de una tupla. Las restricciones de tipo sobre los argumentos de esta regla surgen de forma natural al plantearse las situaciones donde debe aplicarse: ambos argumentos

[3] La cancelación por pasos de los conjuntos coordinantes izquierdo y derecho tiene una motivación lingüística bien conocida, observada por primera vez en [Ross, 1988].

deben ser complementos verbales. Consideraremos un tipo C de complementos verbales, que tiene como subtipos directos np y a vp. Otra posibilidad es que uno de los miembros de la tupla sea, a su vez, una tupla. Por tanto , la tupla ha de ser introducida en la jerarquía como un subtipo directo de C. Esta dependencia jerárquica en la descripción de los objetos lingüísticos permitirá, en las reglas, que se formen n-tuplas.

Por último, necesitamos una regla de *scan* para casar los sintagmas a la izquierda de la conjunción con los elementos de la tupla a la derecha, una vez que ésta ha sido formada. Por lo tanto, la restricción de tipo sobre el argumento derecho de la regla de scan debe ser, de nuevo, C. El segundo argumento debe ser una coordinación de tuplas buscando los elementos que le faltan a su izquierda. Denotaremos a este tipo con el símbolo $c_{\langle\rangle}$. Esta regla tiene que actuar en coordinación con $\mathbf{D}_{\langle\ldots\rangle}$, que será aplicada solamente cuando el último elemento en el conjunto coordinante izquierdo haya sido cancelado.

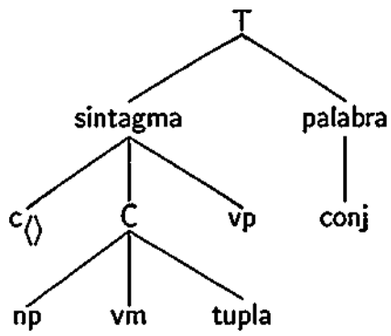
Para establecer la interacción entre las reglas de 7.8 y las reglas parciales, necesitaremos definir algunos operadores entre reglas: un operador de composición, para aplicar sucesivamente varias reglas. Un operador de disyunción, para aplicar varias reglas en paralelo. Y, finalmente, un operador de opcionalidad, para que aplique una regla sólo si da un resultado positivo.

- $r \circ p(x,y) \equiv r(p(x,y))$
- $r \vee p(x,y) \equiv \begin{cases} r(x,y) \text{ si } r(x,y) \neq \perp \text{ y } p(x,y) = \perp \\ p(x,y) \text{ si } p(x,y) \neq \perp \text{ y } r(x,y) = \perp \\ \perp \text{ si } p(x,y) = \perp \text{ y } r(x,y) = \perp \end{cases}$
- $[r](x) \equiv \begin{cases} r(x) \text{ si } r(x) \neq \perp \\ x \text{ si } r(x) = \perp \end{cases}$

En las definiciones anteriores hemos adoptado la convención $\perp$ para simbolizar que la regla no puede ser aplicada. Cuando una regla se aplica sobre $\perp$, el resultado es siempre $\perp$. Cuando una regla genérica retorna el valor $\perp$, entendemos que no se puede combinar la información asociada a los argumentos con que la regla fue invocada.

La jerarquía de objetos lingüísticos $\langle \leq, T \rangle$ que usaremos en la descripción de los fenómenos de coordinación de no constituyentes es:

$$\langle \leq, T \rangle$$



El tipo C representa a los complementos verbales, tal y como los hemos introducido anteriormente. El tipo $c_{\langle\rangle}$ representa la combinación de una conjunción con una tupla como conjunto coordinante derecho. Se necesita para especificar la regla de scan sobre los tipos adecuados de objetos. El tipo tupla se incluye como un complemento verbal para permitir la formación de n-tuplas.

Dada esa jerarquía, proponemos la siguiente regla genérica para analizar oraciones que incluyen fenómenos de coordinación de no constituyentes:

$$\mathcal{SYN} = \begin{cases} SYN_{\mathsf{T}\otimes\mathsf{T}}(\mathsf{X},\mathsf{Y}) = (\mathbf{fa} \vee \mathbf{ba})(\mathsf{X},\mathsf{Y}) \\ SYN_{\mathsf{C}\otimes\mathsf{C}}(\mathsf{X},\mathsf{Y}) = \mathbf{I}_{\langle\ldots\rangle}(\mathsf{X},\mathsf{Y}) \\ SYN_{\mathsf{C}\otimes\mathsf{c}_{\langle\rangle}}(\mathsf{X},\mathsf{Y}) = (\mathbf{D}_{\langle\ldots\rangle} \circ \mathbf{scan})(\mathsf{X},\mathsf{Y}) \end{cases} \tag{7.9}$$

Llamamos de nuevo la atención sobre el hecho de que no ha sido necesario definir ningún tipo de precedencia en la gramática para controlar la interacción de las reglas. Simplemente, un análisis del tipo de argumentos apropiados para cada regla ha provocado, implícitamente, una ordenación parcial entre las distintas reglas parciales. El poset cartesiano asociado a la regla genérica en (7.9) está representado como un diagrama de Hasse en la figura 7.3.

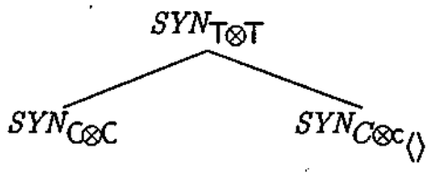


Figura 7.3: Poset cartesiano $\langle \preceq, \mathcal{T}^* \rangle$ asociado al poset $\langle \leq, \mathcal{T} \rangle$ y la regla genérica $\mathcal{SYN}$

Esta regla genérica puede utilizarse para procesar con las mismas técnicas aplicables sobre gramáticas libres de contexto. La única novedad es que hay que mantener un indexado sobre las palabras de la cadena-oración, de forma que la presencia de una conjunción inmediatamente a la izquierda de los argumentos de la regla genérica pueda ser comprobada, para poder aplicar la regla parcial $SYN_{C \otimes C}$.

Veamos un ejemplo de cómo funcionan los procesos de ligadura dinámica asociados a la regla genérica $\mathcal{SYN}$.

Consideremos de nuevo la oración

Juan comió fabada ayer y pisto hoy.

El análisis producido por la regla genérica $\mathcal{SYN}$ puede verse en la figura 7.4. Hemos anotado cada paso de la derivación con la regla parcial aplicada de forma efectiva sobre los sintagmas combinados en cada momento. Esa regla parcial viene dada, en cada caso, por la función de ligadura dinámica **SYN** como la más especifica para combinar los tipos de los argumentos envueltos en el proceso de combinación.

Un parser de gramáticas libres de contexto, con las modificaciones mostradas en la sección 6.1 para el paso de reducción, puede producir éste tipo de análisis a un coste en el tiempo equivalente al de reconocer según una gramática libre de contexto, tanto en la longitud de la cadena como en el tamaño de la gramática. Comparado con el cálculo presentado en la sección 7, la regla genérica es reconocible en tiempo polinómico y puede ser analizada con técnicas de parsing bien conocidas. Además, las restricciones que impone la regla genérica sobre la aplicación de las reglas específicas a través de la asignación de tipos reducen drásticamente el espacio de búsqueda asociado a los procesos de parsing.

Un parser con heurísticas o demonios para controlar los procesos de coordinación podría alcanzar una eficiencia similar. La ventaja indudable de las reglas genéricas es que el conocimiento necesario para reducir el espacio de búsqueda se introduce declarativamente al nivel de la gramática. Este tipo de representación, frente a un parser con heurísticas, tiene una mayor motivación lingüística y mucha mayor incrementalidad y modularidad, tanto para extender la gramática como para el control de los procesos de parsing.

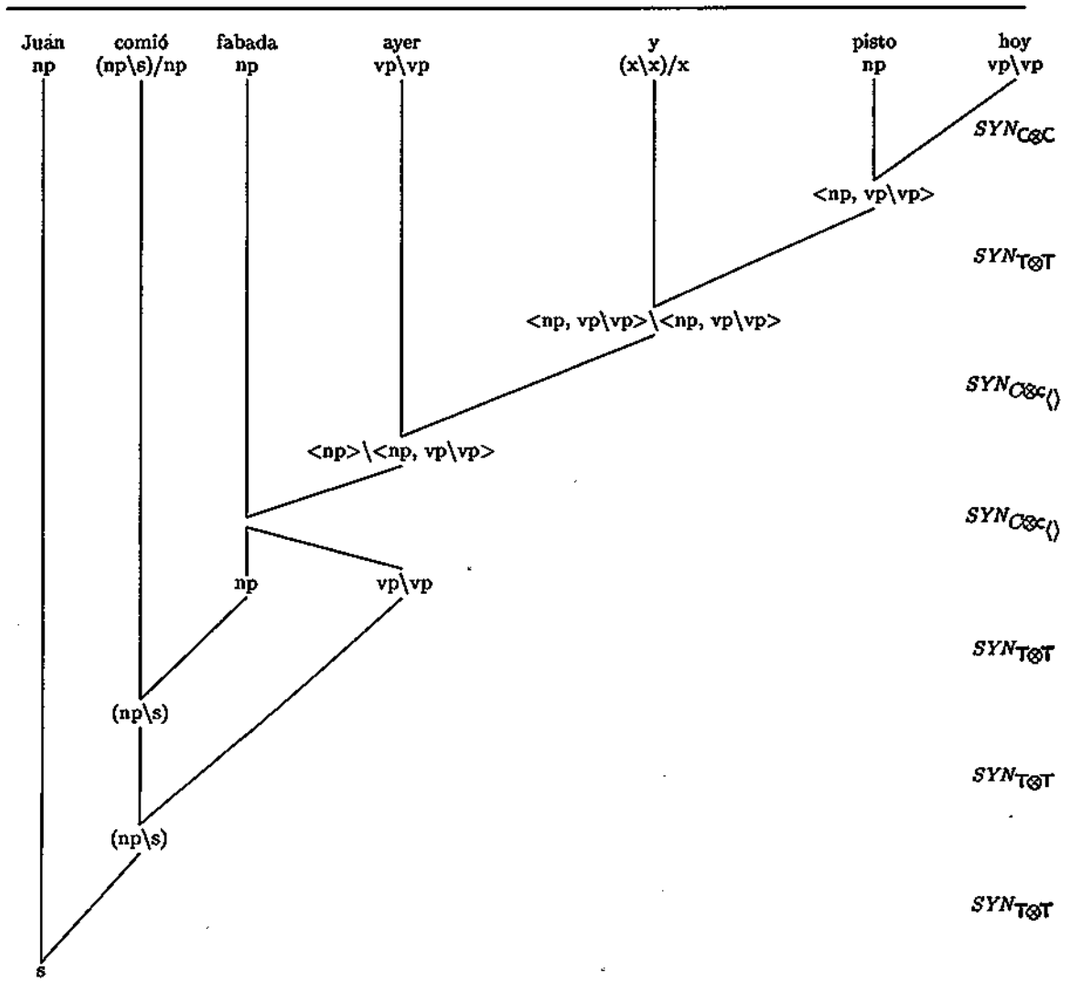


Figura 7.4: Análisis de "Juan comió fabada ayer y pisto hoy" de acuerdo con la regla genérica $\mathcal{SYN}$.

Capítulo 8

Reglas genéricas en el sistema PRÓGENES

El formalismo de reglas genéricas ha sido utilizado en el entorno del sistema PRÓGENES para construir un módulo de interpretación de problemas de matemáticas enunciados en castellano.

PRÓGENES es un sistema capaz de resolver problemas de matemáticas enunciados en lenguaje natural, tal y como los resolvería un estudiante. Una interfaz de lenguaje natural extrae el significado del enunciado en castellano, expresándolo en un lenguaje formal propio del sistema. Un módulo de resolución es capaz de dar respuesta al problema a partir de su expresión formal.

Por ejemplo, si la entrada del sistema es la siguiente:

```
"Hallar la pendiente de la recta $3x+2y+6=0$ y
su intersección con el eje $X$"
```

el analizador de lenguaje natural da lugar a la siguiente expresión formal:

```
(pvalues
 (slope (line $3x+2y+6=0$))
 (intersection
   (line X-axis)
   (line $3x+2y+6=0$)))
```

Esa expresión es la entrada del módulo de resolución. Su salida es otra expresión formal que describe la solución al problema. En este caso:

```
(pvalues 3/2
         (point -2 0))
```

Pueden encontrarse descripciones generales del sistema PRÓGENES en [Castells et al., 1992a, Castells et al., 1992b]. El módulo de resolución de problemas, así como la base de conocimientos, se describen con detalle en [Saiz, 1994, Castells, 1994]. La interfaz de lenguaje natural se presenta en [Díaz and Rodríguez-Marín, 1991, Diaz et al., 1993]. En [Gonzalo et al., 1993] se discuten los problemas lingüísticos relacionados con el dominio. En [Diaz, 1995] se presentan en detalle los fundamentos del sistema de representación semántica y su relación con la base de conocimientos.

El elemento central del sistema, y común a ambos módulos, es una base de conocimiento que describe los elementos del dominio matemático como objetos estructurados con restricciones asociadas. Existe una correspondencia entre los elementos de esta base de conocimiento y las expresiones válidas en el lenguaje formal. El módulo de resolución hace uso de esta correspondencia para resolver el problema. La interfaz de lenguaje natural, por su parte, usa la información de la base de conocimiento para asignar expresiones semánticas al léxico. La interpretación de un enunciado se hace combinando las expresiones semánticas de las palabras que lo componen mediante una gramática compuestas por una regla genérica que relaciona categorías sintácticas y otra que relaciona expresiones semánticas.

La interacción entre reglas por defecto y reglas excepcionales y el desacoplamiento entre sintaxis y semántica que se dan en esta gramática son una buena muestra del uso que puede darse a las reglas genéricas. En este capítulo estudiaremos como se utiliza nuestro formalismo en la interfaz de lenguaje natural de PRÓGENES , después de introducir las características fundamentales de la interfaz.

8.1 Características del dominio

El corpus con el que hemos trabajo se compone de un centenar de exámenes de matemáticas al nivel de primer curso de carreras científicas: biología, física, matemáticas.

El dominio de los enunciados de problemas matemáticos presenta algunas características que lo hacen muy adecuado para una interfaz de lenguaje natural. Así como es poco realista acometer el diseño de un sistema que sea capaz de interpretar semánticamente una conversación de café, un problema enunciado por un matemático es un problema de distinta especie. Al resolver un problema de matemáticas, el lector no debe suponer nada que no se encuentre explícitamente en el texto. En su comprensión no interviene conocimiento pragmático alguno, y el contexto no es otro que el del propio enunciado (de hecho, más de un matemático ilustre aplica también estos principios en sus conversaciones de café, dejando un reguero de anécdotas a sus biógrafos). Los enunciados suelen ser una única oración, de forma que el problema de los referentes del discurso se puede restringir a los

fenómenos intrasentenciales excluyendo sólo una pequeña proporción de problemas.

Otra ventaja desde el punto de vista del procesamiento es que el dominio semántico puede ser formalizado en profundidad sin excesivas dificultades. Los objetos y construcciones matemáticas guardan entre sí relaciones bien definidas. En particular, el conocimiento matemático puede expresarse mediante enunciados de una lógica de primer orden. El dominio propicia, por tanto, la utilización de un formalismo lógico basado en el cálculo λ para realizar la interpretación semántica de los enunciados. Las representaciones lógicas de los procesos semánticos, que resultan aparatosas y excesivamente limitadas en otro tipo de aplicaciones, se ajustan como un guante a este dominio.

El vocabulario, como era de esperar, es muy reducido. Del corpus completo se extraen unas 190 entradas léxicas, incluyendo palabras y expresiones características (como `'desarrollo+de+taylor'`, `"máximo+absoluto"` o `'punto+crítico'`). Sobre un número tan reducido, es innecesario incorporar un sistema de análisis morfológico, de manera que las flexiones que aparecen para cada palabra se incorporan directamente como entradas léxicas independientes.

Una de las características más diferenciadoras del dominio es la presencia de fórmulas matemáticas. En nuestro corpus, las fórmulas aparecen como expresiones TEX entre dólares. Las fórmulas pueden considerarse como léxico dinámico, en el sentido de que sólo existen como entradas léxicas durante el análisis del enunciado del que forman parte. Su papel sintáctico puede ser muy variado. Por ejemplo, en el siguiente problema:

```
Sabiendo que $f$ es derivable en el punto $0$ y que $f'(0)=2$,
calcular $lim_{h \to 0}\frac1h[f(2h)-f(h^2)]$.
```

f actúa sintácticamente como un predicado nominal, e introduce un nuevo referente del discurso. 0 actúa como modificador de un predicado nominal, y $f'(0) = 2$ se comporta como una oración. Por último, $lim_{h\to 0}\frac{1}{h}[f(2h) - f(h^2)]$ es, de nuevo, un predicado nominal. Las dos últimas hacen referencia al referente f introducido en la primera fórmula.

En relación con las fórmulas se descubre pronto que el lenguaje en que se expresan los matemáticos es más ambiguo de lo que parece. Por ejemplo, la fórmula $y = 2x$ tiene tres significados bien distintos en las tres oraciones que siguen:

```
Probar que $y=2x$ es impar

Demostrar que $y=2x$ es par

Hallar la pendiente de  $y=2x$
```

En la primera, $y = 2x$ hace referencia a una función $y(x) = 2x$. y es el nombre de la función, y x es la variable muda que hace referencia a su argumento. Efectivamente, $y(x) = 2x$ es una función impar. En la segunda oración, $y = 2x$ puede hacer referencia a dos variables enteras x e y. Si guardan esa relación, y es siempre un número par. En la tercera oración x e y son sólo variables sintácticas de la ecuación $y = 2x$ que define a una recta.

8.2 Base de Conocimientos y representación semántica

Como hemos dicho, la base de conocimientos constituye el nexo entre la interfaz de lenguaje natural y el módulo de resolución de PRÓGENES . La Base de Conocimientos contiene una descripción declarativa sobre los objetos estructurados del dominio y las relaciones entre ellos. En ella se encuentran cuatro tipos de información:

Objetos Un objeto se define en la base de conocimientos como un tipo de datos. Entre ellos se define una relación de orden que conforma una jerarquía de tipos. Los tipos pueden ser atómicos (como los númeroso reales) o compuestos. Éstos últimos se forman a partir de otros objetos entre los que pueden existir restricciones. Por ejemplo, el objeto circunferencia se define de la siguiente forma:

```
(obj circumference ((center (POINT 2)) (radio REAL))
     (> radio 0))
```

Metafunciones Funciones que transforman unos objetos en otros. La definición de una metafunción contiene un nombre, una especificación de los tipos de sus argumentos y el tipo de su resultado, restricciones que relacionan sus argumentos y una semántica. Además, tiene una condición de aplicación que utiliza el módulo de resolución para priorizar el uso de la metafunción. Por ejemplo, la definición de la función `plano-tangente` es como sigue:

```
(fun (tangent-plane ((s SURFACE) (p (POINT 3)))
      PLANE
     (in p s)
     (plane (normal-vector s p) p)
     (application-condition V)))
```

Esa expresión define dos argumentos para la metafunción `plano-tangente` de tipos `SUPERFICIE` y `PUNTO`. El resultado de la función es un objeto de

tipo plano, que se describe como aquel que tiene como vector dirección un vector normal a la superficie s en el punto p, y que pasa a su vez por el punto p.

Metapredicados Son predicados sobre objetos de la base de conocimientos. Vienen definidos por los mismos elementos que las funciones, con la salvedad de que no se define un tipo del valor que retornan, ya que es siempre un booleano. Por ejemplo, el metapredicado tangente se define como sigue:

```
(pred tangent ((s1 SURFACE) (s2 SURFACE) (p (POINT 3)))
      (pand (in p s1) (in p s2))
      (parallel (normal-vector s1 p) (normal-vector s2 p)))
```

Teoremas Conjunto de reglas que expresan teoremas y propiedades conocidas, y que son utilizados por el módulo de resolución. Por ejemplo:

```
(theor (for-all (<> ?f REALFUN)
         (-> (pand (derivable ?f)
                   (> (der ?f) 0))
                   (strictly-increasing ?f)))
```

Las definiciones de objetos, metafunciones y metapredicados definen un lenguaje formal asociado que llamamos *lenguaje* PRÓGENES . Por ejemplo, la definición del tipo circunferencia lleva asociada la siguiente regla sintáctica:

```
Si <center> cumple TIPO(<center>) < (POINT 2)
 y <radio>  cumple TIPO(<radio>) < REAL
 y (> ?radio 0)
Entonces
 TIPO(circumference <center> <radio> ) := CIRCUMFERENCE
```

Esta regla licencia construcciones como '(circunferencia (punto 2 4) 3)' asignándoles el tipo CIRCUMFERENCE. la definición de la metafunción tangent-plane licencia expresiones como

```
(tangent-plane (sphere (point 0 0 0) 3) (point 3 0 0))
```

Las características de la base de conocimiento y el lenguaje formal asociado se describen con detalle en [Saiz, 1994]. Así como los elementos de la base de conocimiento describen un lenguaje formal asociado equivalente a una lógica de primer orden, se los puede relacionar con las entradas léxicas de la interfaz de lenguaje natural como expresiones equivalentes a un cálculo λ, mediante las que se obtienen expresiones bien formadas en el lenguaje PRÓGENES a través de la aplicación funcional. Esa es la idea básica del sistema de interpretación semántica.

La información que se utiliza para construir esas expresiones equivalentes a un cálculo λ es la que nos da la base de conocimientos sobre la estructura de los objetos, por un lado, y sobre argumentos y tipos de metafunciones y metapredicados, por otro. La diferencia fundamental respecto a un cálculo λ es que cada subexpresión lleva asociada un tipo de la jerarquía de objetos matemáticos del dominio. La aplicación funcional está restringida a la adecuación de los tipos de la variable abstraida en la función y el argumento sobre el que se aplica esa función.

Veamos un par de ejemplos. Del metapredicado tangente

```
(pred tangent ((s1 SURFACE) (s2 SURFACE) (p (PUNTO 3)))
      (pand (in p s1) (in p s2))
      (parallel (normal-vector s1 p) (normal-vector s2 p)))
```

se obtiene la expresión semántica correspondiente a la entrada léxica `tangent`:

```
(BOOL (tangent (SURFACE @s1) (SURFACE @s2) (POINT @p)))
```

Se usa el prefijo '`@`' para denotar las variables semánticas, y el prefijo '`@!`' para las variables semánticas opcionales.

Esta expresión es similar a la que se obtendría mediante abstracción λ del predicado 'tangente' en una lógica de primer orden

$$\lambda x \lambda y \text{TANGENTE}(x, y)$$

La diferencia fundamental es que cada subexpresión lleva asociado un tipo de la jerarquía de objetos del dominio. En particular, las variables semánticas llevan asociado un tipo que debe respetarse al combinar expresiones semánticas mediante aplicación funcional.

De la definición de circunferencia

```
(obj circumference ((center (POINT 2)) (radio REAL))
     (> radio 0))
```

se obtiene la expresión semántica correspondiente a la entrada léxica '`circunferencia`':

```
(CIRCUMFERENCE (circumference (POINT (center (POINT @p)))
                              (REAL  (radio (REAL @r)))))
```

Aunque la funcionalidad dentro del módulo de interpretación semántica es la misma en el caso de la semántica de `tangente` que en el caso de `circunferencia`, el paralelismo de éste último con una expresión de un cálculo λ es menos exacto,

ya que en este caso las variables no responden a los argumentos de un predicado, sino a los componentes de un objeto complejo de tipo CIRCUMFERENCE.

Estas expresiones semánticas, derivadas de la base de conocimientos, constituyen una información indispensable para el proceso de análisis de una oración. De hecho, a partir de las expresiones semánticas asociadas a las palabras de un enunciado, y mediante aplicación funcional, sería posible obtener la expresión formal equivalente a la oración. En la interfaz, se restringen las posibles combinaciones semánticas mediante la estructura sintáctica de las oraciones. A aquellas palabras cuya función es sintáctica, o que no tienen correlato en la base de conocimientos, se les da la semántica '(TYPE @t)', donde TYPE es el tipo más alto de la jerarquía de objetos de la base de conocimientos. Esta semántica es neutra respecto de la operación de combinación semántica que se describe en la sección 8.3.2.

Para conocer en detalle el proceso de extracción semántica de la base de conocimientos puede consultarse [Diaz, 1995].

8.3 Estrategia lingüística

8.3.1 Objetos lingüísticos

Los objetos lingüísticos en PRÓGENES tienen asociados, fundamentalmente, dos tipos de información: sintáctica y semántica. La información sintáctica es una categoría recursiva como las que se utilizan en gramáticas categoriales (ver el capítulo 7). Como categorías básicas se utilizan exclusivamente NP y S. A partir de ellas se construyen, mediante los operadores '/' y '\', las categorías sintácticas de los determinantes ('NP/NP'), verbos intransitivos ('S\NP'), etc.

La información semántica es una expresión obtenida a partir de la base de conocimientos de la forma descrita en el apartado anterior. La representación de los objetos lingüísticos se hace mediante estructuras de rasgos. Los objetos están organizados en una jerarquía de tipos cuya información está relacionada mediante un mecanismo de herencia múltiple por defecto. Este mecanismo es el que se utiliza en CLOS, lenguaje utilizado para la implementación de los objetos lingüísticos. Adaptándonos a las convenciones de CLOS, se distingue entre *clases* (los tipos de la jerarquía) e *instancias* (objetos pertenecientes a una clase). Se adopta la hipótesis de universo abierto: un objeto no ha de ser, necesariamente, una instancia de una clase mínima de la jerarquía, sino que puede ser directamente una instancia de cualquier clase de la jerarquía.

Las estructuras de rasgos que representan a los objetos lingüísticos contienen la información mínima necesaria para el proceso de interpretación. No pretenden ser descripciones lingüísticas completas de las entidades a que hacen referencia.

La clase más alta de la jerarquía es sign. Tiene tres atributos asociados: 'spelling', 'cat' y 'sem'. El primero tiene como valor la cadena asociada a

ese objeto, el segundo la categoría sintáctica y el tercero su expresión semántica asociada. Ninguno de estos elementos de información es, a su vez, una estructura de rasgos, así que el nivel permitido de anidación es cero. Los valores para estos atributos pueden especificarse en la definición de la entrada léxica o heredarse de la clase a la que pertenecen. Veamos un par de ejemplos. La definición de la palabra circunferencia es como sigue:

```
(setf circunferencia
  (make-instance 'noun
    :spelling 'circunferencia
    :sem '(CIRCUMFERENCE (circumference (POINT (center (POINT @p)))
                                (REAL  (radio (REAL @r))))))
```

La información sobre su atributo cat se hereda de la definición de la clase noun:

```
(defclass noun '(word)
  ((cat  :initform 'NP)))
```

de forma que la descripción completa del objeto circunferencia es

```
circunferencia
noun
    :cat 'NP
    :spelling 'circunferencia
    :sem '(CIRCUNFERENCIA (circunferencia (PUNTO (centro (PUNTO @p)))
                                (REAL  (radio (REAL @r))))))
```

La clase sign tiene dos subclases directas: word, a la que pertenecen las entradas léxicas, y phrase, a la que pertenecen los sintagmas creados durante el análisis. Ésta última tiene dos atributos adicionales que sirven para trazar los árboles de análisis y depurar los procesos de combinación de información. El primero, 'daughters', contiene una lista con los dos objetos a partir de los cuáles se ha formado. El segundo, 'rules', contiene la información sobre las reglas usadas para formar el objeto. Por ejemplo:

```
la+pendiente+de+la+recta+formula4+y+su+interseccion+con+el+eje+formula5
U-COORD
 :spelling 'la+pendiente+de+la+recta+formula4+y+su+interseccion+con+el+eje+formula5
 :cat NP
 :sem  '((OBJECT
          (pand (NUM (slope (LINE (line (LINEAR-EQUATION "$y=7x-3$")))))
               (SET (inter (LINE (axis (FORMULA "$X$"))) (SET &SU7))))))
 :daughters (LA+PENDIENTE+DE+LA+RECTA+FORMULA4+Y
             SU+INTERSECCION+CON+EL+EJE+FORMULA5)
 :rules ( "MSYN (sign sign)" "MSEM (pre-coord sign)")
```

Este objeto pertenece a la clase u-coord, que es subclase de phrase. El atributo `rules` hace referencia a las reglas parciales utilizadas para construir la información sintáctica y semántica del objeto. Aclaramos su significado a continuación.

Nuestra representación de los objetos lingüísticos tiene ciertos paralelos con la usada en Unification Categorial Grammar (UCG) [Calder et al., 1988]. La utilización de tipos del dominio de objetos semánticos asociados a las expresiones y subexpresiones semánticas, así como a las variables semánticas, es una de ellas, así como la aproximación categorial a sintaxis y semántica y la minimización del número de rasgos asociados a cada estructura. La diferencia fundamental es que, en UCG, la información sintáctica y semántica está interrelacionada mediante compartición de estructuras, y la Unificación juega un papel fundamental en la combinación de información. En la interfaz de PRÓGENES , sin embargo, no existe una relación explícita entre información sintáctica y semántica. Son tratadas modularmente. Esta modularización permite independizar la descripción de los procesos de combinación como en los ejemplos del capítulo 4.

8.3.2 Reglas gramaticales

El tipo de representación que hemos descrito pone en los objetos lingüísticos casi toda la información necesaria para procesar un enunciado. Sólo hay que postular dos reglas para combinar esos objetos. La primera es la aplicación funcional sobre categorías sintácticas, en sus dos versiones: hacia adelante y hacia detrás:

```
X/Y  Y   -> X  (fa)
Y    X\Y -> X  (ba)
```

La aplicación funcional es la regla básica en una gramática categorial. Otras reglas como la elevación de tipos, la composición funcional, etc... permiten dar cuenta de fenómenos complejos, pero a costa de aumentar exponencialmente el número de análisis posibles y difuminar el concepto de estructura sintáctica. No podemos considerarlos, por tanto, como parte de la gramática básica.

La otra regla es la que gobierna los procesos de combinación semántica. Su funcionalidad es parecida a la aplicación funcional seguida de conversión β en un cálculo λ. Se puede expresar como sigue:

Regla de combinación semántica [app]

$$\begin{array}{l} sem(s_1) = (tipo_1\ f((t_1@v_1),(t_2@v_2),\ldots,(t_n@v_n))) \\ sem(s_2) = (tipo_2\ g) \\ \Longrightarrow \\ sem(s_1 + s_2) = (tipo_1\ f[(t_m\ @v_m)/(tipo_2\ g)]) \end{array} \tag{8.1}$$

donde $(t_m@v_m)$ es la variable semántica más próxima que cumple $t_m \leq tipo_2$.

Es decir, [app] elimina en la semántica del primer objeto aquella variable semántica cuyo tipo es compatible con el de la segunda expresión, sustituyéndola por esa segunda expresión. Si hay más de una, se escoge la más próxima, entendiendo por variable más próxima aquella que se encuentra primero en un recorrido en profundidad de la expresión semántica como lista [1].

La diferencia con la aplicación funcional radica en que las variables semánticas no están ordenadas respecto a la operación de combinación. Esta flexibilidad no ocasiona problemas, porque la jerarquía semántica es muy precisa y determina por si sola la variable adecuada para ser sustituida. Como ventaja, desliga la combinación semántica de la rigidez del orden en el que se descargan las subcategorizaciones en una gramática categorial.

Mediante estas dos reglas para la combinación sintáctica y semántica pueden obtenerse los análisis de un buen número de enunciados que no presentan fenómenos lingüísticos específicos. Como ejemplo, transcribimos aquí una sesión del sistema en la que se procesa el enunciado:

```
"Hallar la interseccion con el eje $y$ de la recta $20x-24y-30=0$"
```

Una vez procesado, el sistema ofrece la posibilidad de navegar por los nodos del árbol de análisis. En la transcripción hacemos uso de esta capacidad para ilustrar los pasos del análisis.

```
USER(6): (parse "Hallar la interseccion con el eje $y$ de la recta $20x-24y-30=0$")

List of signs:
1.- HALLAR+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2
Number (q to quit, r to go back one level)?1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR
            LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2)
Class: PHRASE
Category: S
Semantics (untyped):
((PFIND (INTER (AXIS "$y$") (LINE "$20x-24y-30=0$"))))
ddescribe>1

@@@@@@
HALLAR
@@@@@@

Lexical entry: No daughters
Class: INFINITIVE-VERB
Category: (S / NP)
```

[1] Este criterio es heurístico; en principio, [app] debería devolver tantas posibles soluciones como variables candidatas. Sin embargo, la condición de proximidad para la variable correcta se cumple casi siempre, y reduce un factor de ambigüedad no despreciable en el análisis semántico.

```
Semantics (untyped):
((PFIND @P @!B))
ddescribe>r

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+LA+INTERSECCION+CON+EL+EJE+FORMULAO+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR
            LA+INTERSECCION+CON+EL+EJE+FORMULAO+DE+LA+RECTA+FORMULA2)
Class: PHRASE
Category: S
Semantics (untyped):
((PFIND (INTER (AXIS "$y$") (LINE "$20x-24y-30=0$"))))
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
LA+INTERSECCION+CON+EL+EJE+FORMULAO+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LA+INTERSECCION+CON+EL+EJE+FORMULAO DE+LA+RECTA+FORMULA2)
Class: PHRASE
Category: NP
Semantics (untyped):
((INTER (AXIS "$y$") (LINE "$20x-24y-30=0$"))
 (PEQUAL (INTER (AXIS "$y$") (LINE "$20x-24y-30=0$")) @P))
ddescribe>2

@@@@@@@@@@@@@@@@@@@@
DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@

Daughters: (DE LA+RECTA+FORMULA2)
Class: PP
Category: (NP \ NP)
Semantics (untyped):
((LINE "$20x-24y-30=0$"))
ddescribe>r

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
LA+INTERSECCION+CON+EL+EJE+FORMULAO+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LA+INTERSECCION+CON+EL+EJE+FORMULAO DE+LA+RECTA+FORMULA2)
Class: PHRASE
Category: NP
Semantics (untyped):
((INTER (AXIS "$y$") (LINE "$20x-24y-30=0$"))
 (PEQUAL (INTER (AXIS "$y$") (LINE "$20x-24y-30=0$")) @P))
ddescribe>1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
LA+INTERSECCION+CON+EL+EJE+FORMULAO
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LA+INTERSECCION CON+EL+EJE+FORMULAO)
Class: PHRASE
Category: NP
Semantics (untyped):
((PEQUAL (INTER (AXIS "$y$") @L2) @P) (INTER (AXIS "$y$") @S2))
ddescribe>2

@@@@@@@@@@@@@@@@@@@
CON+EL+EJE+FORMULAO
@@@@@@@@@@@@@@@@@@@
```

```
Daughters: (CON EL+EJE+FORMULA0)
Class: PHRASE
Category: (NP \ NP)
Semantics (untyped):
((AXIS "$y$"))
ddescribe>2

@@@@@@@@@@@@@@@@
EL+EJE+FORMULA0
@@@@@@@@@@@@@@@@

Daughters: (EL+EJE 2&FORMULA0)
Class: PHRASE
Category: NP
Semantics (untyped):
((AXIS "$y$"))
ddescribe>2

@@@@@@@@@@
2&FORMULA0
@@@@@@@@@@

Lexical entry: No daughters
Class: FORMULA
Category: (NP \ NP)
Semantics (untyped):
("$y$")
ddescribe>q
Bye
USER(6):
```

8.4 Regla genérica para la combinación semántica

La formulación original del sistema contemplaba sólo estas dos reglas universales para la combinación sintáctica y semántica. Sin embargo, hay fenómenos característicos de nuestro dominio de los que este planteamiento general no puede dar cuenta. El más frecuente es el de la doble funcionalidad de los determinantes respecto a los nombres que introducen objetos del dominio. Asimismo, existen fenómenos lingüísticos complejos que no pueden resolverse fácilmente disponiendo únicamente de la aplicación funcional. Es el caso de las propiedades distributivas de la coordinación respecto de las expresiones semánticas, o de las referencias pronominales.

El formalismo de reglas genéricas nos proporciona, a nuestro entender, la forma más natural de introducir nuevas reglas que den cuenta de estos fenómenos.

Las reglas de aplicación pueden introducirse como aquellas reglas parciales aplicables al combinar objetos del tipo más inespecífico, sign:

$$\mathcal{SEM} = \left\{ \; SEM_{\mathsf{sign}\otimes\mathsf{sign}}(\mathrm{X},\mathrm{Y}) = [\mathrm{app}](sem(\mathrm{X}), sem(\mathrm{Y})) \right. \tag{8.2}$$

$$\mathcal{SYN} = \{\ SYN_{\text{sign}\otimes\text{sign}}(\mathsf{X},\mathsf{Y}) = (\text{fa}\ \vee \text{ba}\ )(cat(\mathsf{X}), cat(\mathsf{Y})) \qquad (8.3)$$

donde el operador de disyunción $\vee$ es el definido en el capítulo 7.

A continuación vamos a dar un repaso a las reglas parciales más significativas del sistema que dan cuenta de fenómenos semánticos que no pueden ser tratados mediante aplicación funcional.

8.4.1 Determinantes y asignación de tipos

Así como en lógica de primer orden las clases de objetos se suelen representar mediante predicados de aridad 1, en el lenguaje PRÓGENES tienen sintácticamente un comportamiento distinto. Además de actuar como constructores, como en el ejemplo anterior, los nombres de objetos pueden funcionar semánticamente como asignaciones de tipo. En el siguiente problema se dan las dos situaciones:

```
"Encontrar el plano perpendicular a la recta $2x-2y+z+1=0,
x+2y+2z-1=0$ que pasa por el punto $(-1 0 3)$"
```

Su traducción al lenguaje formal PRÓGENES es la siguiente:

```
(pfind
  (pthe (: %P plano)
          (perpendicular
             %P
             (line "$2x-2y+z+1=0,x+2y+2z-1=0$"))
          (pertenece (point "$(-1 0 3)$")
                     %P)))
```

El nombre `plano` no se traduce como un constructor ni como un predicado, sino como una asignación de tipo sobre una variable que introduce el determinante. Esto se debe a que los nombres de objetos son en la base de conocimiento una jerarquía de tipos, así que aparecen en el lenguaje formal como asignación de tipo y no como predicado. En el caso de `recta` y `punto`, sin embargo, introducen los constructores correspondientes del lenguaje formal, que se comportan sintácticamente como predicados.

No se puede dar cuenta del funcionamiento como asignación de tipo de los nombres mediante la aplicación funcional, ya que la aplicación funcional se hace sobre variables tipadas, pero no puede hacerse exclusivamente sobre un tipo.

Estos dos casos corresponden a los dos posibles efectos cuantificadores del determinante. Para dar cuenta de ellos, definiremos una nueva regla que combine determinantes con nombres, dando las dos posibilidades semánticas:

Regla de cuantificación del nombre [quant]

```
(OBJ sem)
-->
(OBJ sem)
(OBJ (pthe (OBJ (: (OBJ %X) (OBJ obj)))
           (BOOL @b!)))
```

En la primera expresión resultante, el nombre funciona como constructor y la acción del determinante es neutra. En la segunda, el determinante cuantifica sobre una nueva variable '%X'.Nótese que se trata de una variable del lenguaje formal introducida por un predicado de asignación de tipos, y no tiene nada que ver con las variables semánticas que intervienen en el proceso de análisis, y que deben estar ausentes de la expresión final que se pasa al módulo de resolución.

Mediante [quant] podemos definir la siguiente regla parcial:

$$SEM_{\text{determiner}\otimes\text{noun}}(X, Y) = [\text{quant}](sem(Y)) \tag{8.4}$$

que da cuenta correctamente del comportamiento de los determinantes en el lenguaje formal PRÓGENES .

Durante el análisis, aquellos predicados que involucren una variable introducida por un determinante dejarán, al final del análisis, una variable semántica sin resolver cuyo tipo es compatible con el de esa variable. Para resolver estas referencias (asociadas en el texto a pronombres anáforicos como "que" o "donde"), al final del análisis se aplica un algoritmo de resolución de referentes, que sustituye cada variable semántica pendiente por la variable que introduce su entorno más cercano. Se pueden ver los detalles de este proceso en [Diaz, 1995].

8.4.2 Coordinación de constituyentes

La coordinación de constituyentes no constituye un problema desde el punto de vista sintáctico. Basta con asignar la categoría polimórfica '(X/X)\X' a las conjunciones para que pueda coordinar objetos de cualquier categoría, siempre que el sintagma a la izquierda de la conjunción tenga la misma categoría que el sintagma a la derecha (ver el capítulo 7).

Sin embargo, desde el punto de vista semántico ofrece ciertas peculiaridades que, de nuevo, no pueden ser tratadas simplemente mediante aplicación funcional. Veamos el siguiente ejemplo:

```
"Hallar la pendiente y la intersección con el eje $y$
de la recta $20x-24y-39=0$"
```

que tiene la siguiente traducción en lenguaje PRÓGENES :

```
(pvalues (pfind (slope (line "$20x-24y-39=0$")))
         (pfind (inter (axis "$y$") (line "$20x-24y-39=0$"))))
```

Las coordinaciones tienen propiedades distributivas respecto de la aplicación semántica. En el ejemplo, 'la pendiente y la intersección con el eje y' se enlaza distributivamente con la semántica de 'de la recta $20x-24y-39=0$'. De nuevo, el conjunto se enlaza distributivamente a su izquierda con la semántica de hallar. Este tipo de distributividad no puede, sin embargo, aplicarse indiscriminadamente. Por ejemplo, en el siguiente enunciado:

```
"Encontrar la recta que pasa por el punto $(3 2)$ y el punto $(4 1)$"
```

según el alcance semántico que se dé a la coordinación, serían posibles hasta tres traducciones distintas:

(A)

```
(pfind (pthe (: %X line)
          (in %X (pand (point $(3 2)$)
                       (point $(4 1)$)))))
```

(B)

```
(pfind (pthe (: %X line)
          (pand (in %X (point $(3 2)$))
                (in %X (point $(4 1)$)))))
```

(C)

```
(pvalues (pfind (: %X line)
             (in %X (point $(3 2)$)))
         (pfind (: %X line)
             (in %X (point $(4 1)$))))
```

De esos tres análisis, sólo (B) es correcto. El primero es descartable porque el predicado de pertenencia no puede tener como segundo argumento una lista de puntos. El tercero, sin embargo, es una expresión correcta del lenguaje formal PRÓGENES , y podría corresponder a la oración

```
"Encontrar la recta que pasa por $(3 2)$ y encontrar
la recta que pasa por $(4 1)$"
```

aunque el módulo de resolución no sería capaz de encontrar solución única a cada uno de estos dos problemas.

Este ejemplo llama la atención sobre algunas de las dificultades que se plantean al analizar estructuras coordinadas. Se necesita una gramática para la coordinación de constituyentes que cumpla tres requisitos: uno, que de cuenta de las diferencias entre coordinación sintáctica y coordinación semántica. Dos, que produzca el mínimo número de análisis espurios (en el ejemplo anterior, que sólo produzca la segunda expresión semántica). Y tres, que no interfiera con el funcionamiento del resto de la gramática.

Para ello, en PRÓGENES hemos definido varias reglas parciales semánticas que aplican la distributividad semántica según un principio de *coordinación completa*.

En el lenguaje formal hay dos tipos de expresiones relacionadas con la conjunción: 'pand', que equivale al operador lógico de conjunción '$\wedge$', y coordina, por tanto, expresiones de tipo `BOOL`; y 'pvalues', que es una función de control con un funcionamiento similar al `progn` de Lisp. Es decir, produce la evaluación de cada uno de los elementos coordinados. Para analizar correctamente este último tipo de construcciones, se introduce en la interfaz de lenguaje natural el tipo `ACTION` para las expresiones de funciones de control como `pfind` o `prove`. Este tipo sólo tiene significado a efectos de combinación semántica, y no es considerado por el sistema de resolución.

Así como en una lógica de primer orden una conjunción siempre relaciona predicados, en el lenguaje formal PRÓGENES una conjunción relaciona usualmente proposiciones, es decir, sintagmas cuyo valor semántico es una expresión de tipo `BOOL` o de tipo `ACTION`. Durante el proceso de análisis de un enunciado, sin embargo, podemos estar manejando sintagmas que coordinan expresiones semánticas de otros tipos, como en el caso de '(32) y (41)'. Llamaremos *incompletas* a este tipo de coordinaciones, y *completas* a aquellas coordinaciones de tipo `BOOL` o `ACTION`.

Adoptaremos el siguiente principio: *El alcance de una conjunción será el de la mínima coordinación completa.* De acuerdo con este principio, la única traducción correcta de la oración anterior es `(B)`. En `(A)`, la coordinación es incompleta. En `(C)`, la coordinación es completa, pero no mínima.

Para que las conjunciones tomen su alcance adecuado en la interpretación de un enunciado, las coordinaciones incompletas presentarán propiedad distributiva respecto a su combinación a la izquierda. Es decir, al combinar una coordinación incompleta con un constituyente a su izquierda, obtendremos una nueva coordinación cuyos elementos resultarán de la aplicación funcional de la expresión semántica de la izquierda sobre cada una de las expresiones coordinadas.

Por otro lado, las coordinaciones, completas o incompletas, presentan la propiedad de distribución semántica por la derecha. Es decir, al combinar una coordinación con un constituyente a su derecha, la semántica de éste se insertará en cada uno de los miembros coordinados.

Además de las características distributivas de las coordinaciones, hay que tener en cuenta otro comportamiento excepcional: la asignación de tipo semántico a una coordinación durante el proceso de análisis. En la aplicación funcional normal, el tipo de la expresión resultante es el mismo que el de la expresión semántica que hace de functor. Al coordinar dos expresiones semánticas, sin embargo, el tipo debe ser el mínimo supertipo común de las dos.

Por último, al alcanzarse una coordinación completa hay que utilizar el término PRÓGENES adecuado para la conjunción: pand en caso de que se coordinen expresiones booleanas o pvalues en caso de que se combinen expresiones de tipo `ACTION`.

Para codificar este comportamiento mediante reglas parciales, introduciremos varias clases nuevas en la jerarquía lingüística: coord, u-coord y pre-coord. La primera, coord, es la clase a la que pertenecen todas las coordinaciones. u-coord es subclase de coord, y pertenecen a ella aquellas coordinaciones incompletas. La tercera es la clase de los objetos que resultan de combinar una conjunción con algún signo a su derecha. Puede verse como se introducen en la jerarquía en la figura 8.1.

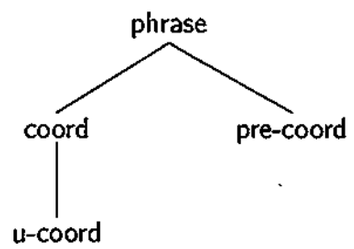


Figura 8.1: Clases relacionadas con la coordinación

Describiremos el comportamiento de las coordinaciones mediante tres reglas. La primera asigna el tipo a la coordinación:

Regla de asignación de tipo para la coordinación [join]

```
(T1 sem1)
(TYPE (conj (T2 sem2)
            (TYPE @t)))
-->
(J (conj (T2 sem2)
         (T1 sem1)))
```

donde J es el mínimo supertipo común de T2 y T1, y conj debe sustituirse por pand si J $\leq$ BOOL, y por pvalues si J $\leq$ ACTION.

La regla parcial que recoge este funcionamiento debe asociarse a un primer argumento inespecífico, es decir, de tipo sign, y a un segundo argumento de tipo pre-coord. Queda, pues, como sigue:

$$SEM_{\text{sign}\otimes\text{pre-coord}}(X, Y) = [\text{join}](sem(X), sem(Y)) \tag{8.5}$$

La segunda describe la distributividad de las coordinaciones a su derecha:

Regla de distributividad de la coordinación por la derecha [dist→]

```
(J (conj (T1 sem1)
         (T2 sem2)))
(T3 sem3)
-->
(J (conj [app]((T1 sem1),(T3 sem3))
         [app]((T2 sem2),(T3 sem3))))
```

La regla parcial que recoge este funcionamiento debe tener como argumentos un objeto coord y un objeto sign. Podemos, pues, escribirla así:

$$SEM_{\text{coord}\otimes\text{sign}}(X, Y) = [\text{dist}\rightarrow](sem(X), sem(Y)) \tag{8.6}$$

La tercera aplica la distributividad a la izquierda de las coordinaciones incompletas, es decir, aquellas de clase u-coord:

Distributividad de la coordinación por la izquierda [dist←]

```
(T3 sem3)
(J (conj (T1 sem1)
         (T2 sem2)))
-->
(t3 (conj [app]((T3 s3),(T1 s1))
          [app]((T3 s3),(T2 s2))))
```

La regla parcial que hace uso de esta propiedad debe establecerse sobre objetos de tipo sign por la izquierda y coordinaciones incompletas, es decir, de tipo u-coord, por la derecha. La definimos, por tanto, de esta forma:

$$SEM_{\text{sign}\otimes\text{u-coord}}(X, Y) = [\text{dist}\leftarrow](sem(X), sem(Y)) \tag{8.7}$$

Estas tres reglas dan cuenta de los fenómenos semánticos en las oraciones que hemos considerado y, en general, de la mayoría de los fenómenos de coordinación

en nuestro corpus. Para generalizar su funcionamiento a coordinaciones de n elementos, hay que modificar trivialmente la definición de [join], [dist→] y [dist←]. Además, hay que añadir una regla para la combinación de una conjunción con una coordinación:

Ampliación de la coordinación [add-conj]

```
(J (conj (T1 sem1)
         (T2 sem2)
          ...
         (Tn semn)))
-->
(J (conj (T1 sem1)
         (T2 sem2)
          ...
         (Tn semn)
         (TYPE @t)))
```

que debe asociarse a una regla parcial con tipo cartesiano conj⊗coord:

$$SEM_{\text{conj}\otimes\text{coord}}(X, Y) = [\text{add-conj}](sem(Y)) \tag{8.8}$$

Al introducir esta regla parcial en el sistema se señala un error de mala formación (ver el capítulo 5). El poset cartesiano asociado a la regla genérica está mal formado porque puede producirse un conflicto de precedencia si aplicamos la regla genérica sobre dos argumentos que formen el tipo cartesiano conj⊗u-coord. En ese caso, no hay forma de decidir si la regla más específica es $SEM_{\text{conj}\otimes\text{coord}}$ o $SEM_{\text{sign}\otimes\text{u-coord}}$. Esta indeterminación se ilustra en la figura 8.2.

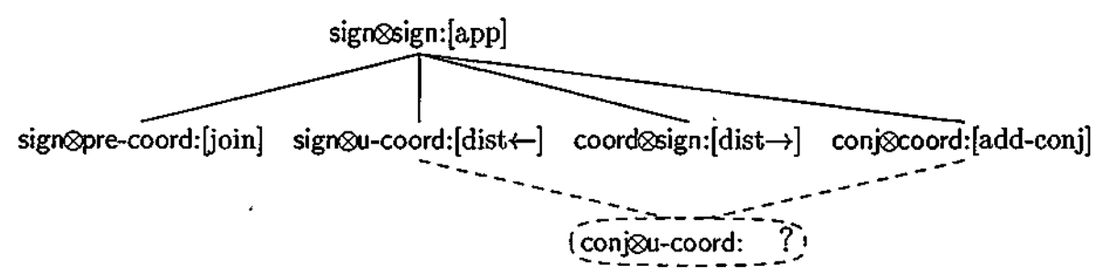


Figura 8.2: Si los argumentos forman el tipo conj⊗u-coord, no puede establecerse la regla más específica.

Para solucionar una malformación hay que incluir una nueva regla parcial cuyo tipo cartesiano sea el formado por el mayor subtipo común de sign y conj, a la izquierda, y el mayor subtipo común de u-coord y coord, a la derecha. En este caso, ese tipo es precisamente conj⊗u-coord. Es obvio que la regla que necesitamos al combinar conjunciones con coordinaciones, ya sean completas o no, es la recién definida [add-conj]. Por tanto, la regla parcial que soluciona la malformación es:

$$SEM_{\text{conj}\otimes\text{u-coord}}(X, Y) = [\text{add-conj}](sem(Y)) \tag{8.9}$$

Introducir la información adicional que necesita el sistema a través de una nueva regla tiene la ventaja de que no es necesario definir ningún mecanismo adicional de ordenación, al estilo de las listas de precedencia de clases mediante las que CLOS resuelve los conflictos de herencia múltiple por defecto. Como contrapartida, no distinguimos entre las reglas parciales con significado lingüístico y las que simplemente solucionan problemas de precedencia.

El principio de coordinación mínima que hemos usado para escribir este sistema de reglas tiene un contraejemplo principal: el de los *predicados selectivos*. En nuestro caso, aquellas funciones del lenguaje formal con varios argumentos del mismo tipo, que son normalmente invariantes respecto al orden de los argumentos. Mientras que oraciones como

```
"Hallar la derivada de $f=(x^2-1)/x$ y $g=\sin(x^2)$"

(pvalues (pfind (der $f=(x^2-1)/x$))
         (pfind (der $g=\sin(x^2)$)))

"Calcular la pendiente de $3x-2y+5=0$ y $-7x+y+5=0$"

(pvalues (pfind (slope $3x-2y+5=0$))
         (pfind (slope $f=(x^2-1)/x$)))
```

son analizadas correctamente, estas otras, aparentemente idénticas:

```
"Hallar la convolución de $f=(x^2-1)/x$ y $g=\sin(x^2)$"

(pfind (convolution $f=(x^2-1)/x$
                    $g=\sin(x^2)$))

"Calcular el punto de intersección de $3x-2y+5=0$ y $-7x+y+5=0$"

(pfind (intersection-point $3x-2y+5=0$
                           $-7x+y+5=0$))
```

se comportan semánticamente de un modo totalmente distinto al que dictan las

reglas parciales introducidas hasta ahora. Cada uno de los elementos coordinados pasa a ser un argumento distinto de la metafunción

```
(REALFUN (convolution (REALFUN @r1) (REALFUN @r2)))
```

en el primer caso, y de la metafunción

```
(POINT (intersection-point (LINE @l1)
                           (LINE @l2)))
```

en el segundo.

La forma menos comprometida de localizarlo es suponer que se comportan así las coordinaciones en presencia de aquellos objetos cuya interpretación semántica es un constructor con dos o más argumentos del mismo tipo. Nótese que no es un requisito el que los elementos coordinados sean del mismo tipo. Un ejemplo es:

```
"Hallar la intersección de la recta $x+y+z=1, x-y+z=3$
y el plano $3x+2y+z=2$"
```

donde la semántica de intersección es '`(SET (intersection (SET @s1) (SET @s2)))`', pero recta y plano tienen tipos distintos `LINE`, `PLANE`, ambos subtipos de `SET`.

El formalismo de reglas genéricas nos permite estudiar esta situación sin necesidad de revisar las reglas parciales que hemos incorporado ya a la gramática. Primero necesitamos una nueva regla que exprese ese comportamiento:

[split]

```
(T1 f((T2 @s1),(T2 @s2)))
(J (conj (T3 sem3)
         (T4 sem4)))
-->
(T1 f[(T2 @s1)/(T3 sem3),(T2 @s2)/(T4 sem4)])
```

donde T3 $\leq$ T2 y T4 $\leq$ T2.

Si consideramos una nueva clase de objetos lingüísticos sym que describe a aquellos objetos cuya interpretación semántica es un constructor con dos o más argumentos del mismo tipo, entonces la regla parcial se define así:

$$SEM_{\text{sym}\otimes\text{u-coord}}(\mathsf{X},\mathsf{Y}) = [\text{split}](sem(\mathsf{X}), sem(\mathsf{Y})) \tag{8.10}$$

Esta regla representa un comportamiento excepcional respecto a la regla distributiva general $SEM_{\text{sign}\otimes\text{u-coord}}$. De esta forma, hemos dado cuenta de fenómenos que alteran el tratamiento general de forma modular e incremental.

A continuación reproducimos una sesión en la que se procesa el enunciado

```
"Hallar la pendiente y la interseccion con el eje $y$
de la recta $20x-24y-39=0$"
```

Como en ocasiones anteriores, paseamos por la estructura del análisis, una vez formado, para mostrar el proceso de composición semántica.

```
USER(2): (parse "Hallar la pendiente y la interseccion con el eje $y$
de la recta $20x-24y-39=0$")
List of signs:
1.- HALLAR+LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2
Number (q to quit, r to go back one level)?1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR
            2&LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2)
Class: COORDINATION
Category: S
Semantics (untyped):
((PVALUES (PFIND (SLOPE (LINE "$20x-24y-39=0$")))
  (PFIND (INTER (AXIS "$y$") (LINE "$20x-24y-39=0$")))))
ddescribe>sem

sem:
((ACTION
  (PVALUES
   (ACTION
    (PFIND
     (NUM (SLOPE (LINE (LINE (LINEAR-EQUATION "$20x-24y-39=0$")))))))
   (ACTION
    (PFIND
     (SET (INTER (LINE (AXIS (FORMULA "$y$")))
           (LINE (LINE (LINEAR-EQUATION "$20x-24y-39=0$"))))))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (sign u-coord)")
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
2&LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULA0+DE+LA+RECTA+FORMULA2
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULA0
            DE+LA+RECTA+FORMULA2)
Class: U-COORD
Category: NP
Semantics (untyped):
((AND (SLOPE (LINE "$20x-24y-39=0$"))
      (PEQUAL (INTER (AXIS "$y$") (LINE "$20x-24y-39=0$")) @P))
 (AND (SLOPE (LINE "$20x-24y-39=0$"))
      (INTER (AXIS "$y$") (LINE "$20x-24y-39=0$"))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (coordination sign)")
ddescribe>1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
LA+PENDIENTE+Y+LA+INTERSECCION+CON+EL+EJE+FORMULAO
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LA+PENDIENTE+Y LA+INTERSECCION+CON+EL+EJE+FORMULAO)
Class: U-COORD
Category: NP
Semantics (untyped):
((AND (SLOPE @R) (INTER (AXIS "$y$") @S2))
 (AND (SLOPE @R) (PEQUAL (INTER (AXIS "$y$") @L2) @P))
 (AND (PTHE (<> %X NUM) @B) (INTER (AXIS "$y$") @S2))
 (AND (PTHE (<> %X NUM) @B) (PEQUAL (INTER (AXIS "$y$") @L2) @P)))
ddescribe>sem

sem:
((OBJECT
  (AND (NUM (SLOPE (LINE @R)))
       (SET (INTER (LINE (AXIS (FORMULA "$y$"))) (SET @S2)))))
 (TYPE
  (AND (NUM (SLOPE (LINE @R)))
       (BOOL
        (PEQUAL
         (POINT (INTER (LINE (AXIS (FORMULA "$y$"))) (LINE @L2)))
         (POINT @P)))))
 (OBJECT
  (AND (NUM (PTHE (TYPE-ASSIGN (<> (VAR %X) (NUM NUM))) (BOOL @B)))
       (SET (INTER (LINE (AXIS (FORMULA "$y$"))) (SET @S2)))))
 (TYPE
  (AND (NUM (PTHE (TYPE-ASSIGN (<> (VAR %X) (NUM NUM))) (BOOL @B)))
       (BOOL
        (PEQUAL
         (POINT (INTER (LINE (AXIS (FORMULA "$y$"))) (LINE @L2)))
         (POINT @P))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)" "MSEM :primary (pre-coord sign)")
ddescribe>q
Bye
USER(3):
```

8.4.3 Posesivos

Como hemos mencionado, la mayoría de los enunciados están formados por una sola oración, de forma que los fenómenomenos relacionados con el discurso, como las referencias pronominales, son escasos. Aunque no hemos desarrollado un tratamiento serio para ellos, si hemos tenido en cuenta el más frecuente, que es el del posesivo "su" apareciendo en oraciones en las que hay más de una proposición coordinada en el nivel semántico, como en el ejemplo que abría este capítulo:

"Hallar la pendiente de la recta $3x + 2y + 6 = 0$ y su intersección con el eje X"

Para resolverlos, hemos incorporado al sistema una regla parcial que marca las variables semánticas afectadas por el pronombre posesivo. La forma de marcarlas es renombrarlas en la forma '&SUn', donde n es la numeración que les asigna el proceso de marcaje. Este proceso se hace mediante la siguiente regla:

Regla de posesivo [pos]

```
(T f(sem1,sem2,...,semk))
-->
(T f[sem1/&SUn,sem2/&SUn,...semk/&SUn])
```

donde n es un número tal que &SUn no ha sido utilizada previamente en el proceso de análisis.

La regla parcial que recoge este comportamiento es:

$$SEM_{\text{possessive}\otimes\text{sign}}(X, Y) = [\text{pos}](sem(Y)) \tag{8.11}$$

Para el enunciado que hemos tomado como ejemplo, la regla actúa de la siguiente forma:

```
intersección+con+el+eje+$x$: (SET (intersection (AXIS $x$) (SET @s)))
-->
su+intersección+con+el+eje+$x$: (SET (intersection (AXIS $x$) (SET &SU1)))
```

Esas variables son instanciadas al final del análisis, siguiendo el siguiente criterio: cada variable &SUn será sustituida por la subexpresión de tipo compatible más específico en el primer elemento coordinante. Dos variables con el mismo nombre han de ser sustituidas por expresiones distintas y del mismo tipo.

Por ejemplo, para la expresión final del proceso de análisis:

```
(ACTION (pvalues
          (ACTION (pfind
                    (NUM (slope
                           (LINE (line
                                   (LINEAR-EQUATION $3x+2y+6=0$)))))))
          (ACTION (pfind
                    (SET (intersection (AXIS $x$) (SET &SU1)))))))
```

el proceso de resolución de referentes de posesivo sustituye &SU1 por '(LINE (line (LINEAR-EQUATION $3x+2y+6=0$)))', que en este caso es la única subexpresión en el primer elemento coordinante cuyo tipo es compatible con SET.

A continuación transcribimos una sesión en la que se analiza el enunciado:

```
"Hallar el punto donde se intersectan las rectas $y=3x-2$
y $Y=x-7$ y determinar su angulo de interseccion"
```

Se trata de un ejemplo relativamente complejo, en el que intervienen dos coordinaciones anidadas y buena parte de las reglas descritas en esta sección. Una vez analizada, en la sesión navegamos por la estructura del enunciado ya procesado para mostrar cómo han interaccionado las distintas reglas semánticas.

```
USER(4): (parse "Hallar el punto donde se intersectan las rectas
$y=3x-2$ y $Y=x-7$ y determinar su angulo de interseccion")
List of signs:
1.- HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1+Y+DETERMINAR+SU+ANGULO+DE+INTERSECCION
Number (q to quit, r to go back one level)?1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1+Y+DETERMINAR+SU+ANGULO+DE+INTERSECCION
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1+Y
           DETERMINAR+SU+ANGULO+DE+INTERSECCION)
Class: COORDINATION
Category: S
Semantics (untyped):
((PVALUES (PFIND (INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")))
  (PFIND (ANGLE "y=3x-2$" "Y=x-7$"))))
ddescribe>sem

sem:
((ACTION
  (PVALUES
   (ACTION
    (PFIND
     (SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                        (LINE (LINE (LINEAR-EQUATION "Y=x-7$"))))))
   (ACTION
    (PFIND
     (NUM
      (ANGLE (LINEAR-EQUATION "y=3x-2$")
       (LINEAR-EQUATION "Y=x-7$")))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)" "MSEM :primary (pre-coord sign)")
ddescribe>1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1+Y
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1
           Y)
Class: PRE-COORD
Category: (S / S)
Semantics (untyped):
((AND (PFIND (INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")) @!B) @T))
ddescribe>sem

sem:
((TYPE
  (AND (ACTION
        (PFIND
         (SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                            (LINE (LINE (LINEAR-EQUATION "Y=x-7$")))))
         (BOOL @!B)))
       (TYPE @T))))
ddescribe>1

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
HALLAR+EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (HALLAR
            EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULA0+Y+FORMULA1)
Class: PHRASE
```

```
Category: S
Semantics (untyped):
((PFIND (INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")) @!B))
ddescribe>sem

sem:
((ACTION
  (PFIND
   (SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                      (LINE (LINE (LINEAR-EQUATION "Y=x-7$")))))
   (BOOL @!B))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (sign sign)")
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
EL+PUNTO+DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULAO+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (EL+PUNTO
            DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULAO+Y+FORMULA1)
Class: PHRASE
Category: NP
Semantics (untyped):
((INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")))
ddescribe>sem

sem:
((SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                    (LINE (LINE (LINEAR-EQUATION "Y=x-7$"))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (determiner-noun phrase)" )
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
DONDE+SE+INTERSECTAN+LAS+RECTAS+FORMULAO+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (DONDE SE+INTERSECTAN+LAS+RECTAS+FORMULAO+Y+FORMULA1)
Class: PHRASE
Category: (NP \ NP)
Semantics (untyped):
((INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")))
ddescribe>sem

sem:
((SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                    (LINE (LINE (LINEAR-EQUATION "Y=x-7$"))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (sign sign)" )

ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
SE+INTERSECTAN+LAS+RECTAS+FORMULAO+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
Daughters: (SE+INTERSECTAN LAS+RECTAS+FORMULA0+Y+FORMULA1)
Class: PHRASE
Category: (S \ NP)
Semantics (untyped):
((INTERSECTION (LINE "y=3x-2$") (LINE "Y=x-7$")))
ddescribe>sem

sem:
((SET (INTERSECTION (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
                    (LINE (LINE (LINEAR-EQUATION "Y=x-7$"))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (sym coordination)")
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
LAS+RECTAS+FORMULA0+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

Daughters: (LAS+RECTAS 2&FORMULA0+Y+FORMULA1)
Class: U-COORD
Category: NP
Semantics (untyped):
((AND (LINE "y=3x-2$") (LINE "Y=x-7$")))
ddescribe>sem

sem:
((LINE
  (AND (LINE (LINE (LINEAR-EQUATION "y=3x-2$")))
       (LINE (LINE (LINEAR-EQUATION "Y=x-7$"))))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)"
 "MSEM :primary (sign u-coord)")
ddescribe>2

@@@@@@@@@@@@@@@@@@@@@@
2&FORMULA0+Y+FORMULA1
@@@@@@@@@@@@@@@@@@@@@@

Daughters: (2&FORMULA0+Y 2&FORMULA1)
Class: U-COORD
Category: (NP \ NP)
Semantics (untyped):
((AND "y=3x-2$" "Y=x-7$"))
ddescribe>sem

sem:
((FORMULA (AND (FORMULA "y=3x-2$") (FORMULA "Y=x-7$"))))
ddescribe>rules

rules used to create the object:
( "MSYN :primary (sign sign)" "MSEM :primary (pre-coord sign)")
ddescribe>q
Bye
USER(5):
```

8.5 Evaluación

La regla genérica que regula la combinación semántica en la interfaz de lenguaje natural de PRÓGENES queda conformada, según las reglas parciales que hemos descrito, de esta forma:

$$\mathcal{SEM} = \begin{cases} SEM_{\text{sign}\otimes\text{sign}}(\text{X},\text{Y}) = [\text{app}](sem(\text{X}), sem(\text{Y})) \\ SEM_{\text{determiner}\otimes\text{noun}}(\text{X},\text{Y}) = [\text{quant}](sem(\text{Y})) \\ SEM_{\text{sign}\otimes\text{pre-coord}}(\text{X},\text{Y}) = [\text{join}](sem(\text{X}), sem(\text{Y})) \\ SEM_{\text{coord}\otimes\text{sign}}(\text{X},\text{Y}) = [\text{dist}\rightarrow](sem(\text{X}), sem(\text{Y})) \\ SEM_{\text{sign}\otimes\text{u-coord}}(\text{X},\text{Y}) = [\text{dist}\leftarrow](sem(\text{X}), sem(\text{Y})) \\ SEM_{\text{conj}\otimes\text{coord}}(\text{X},\text{Y}) = [\text{add-conj}](sem(\text{Y})) \\ SEM_{\text{conj}\otimes\text{u-coord}}(\text{X},\text{Y}) = [\text{add-conj}](sem(\text{Y})) \\ SEM_{\text{sym}\otimes\text{u-coord}}(\text{X},\text{Y}) = [\text{split}](sem(\text{X}), sem(\text{Y})) \\ SEM_{\text{possessive}\otimes\text{sign}}(\text{X},\text{Y}) = [\text{pos}](sem(\text{Y})) \end{cases} \tag{8.12}$$

En la figura 8.3 se representa el diagrama de Hasse para esta regla genérica.

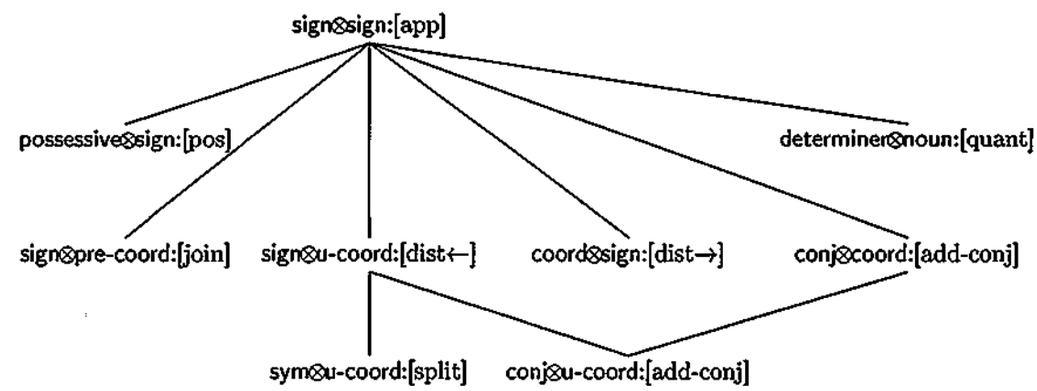


Figura 8.3: Diagrama de Hasse asociado a la regla genérica $\mathcal{SEM}$

El desarrollo de esta gramática para la combinación semántica en PRÓGENES pone de manifiesto las ventajas del formalismo de reglas genéricas. Hemos partido de un mecanismo general, [app], que da cuenta de la combinación semántica. Aunque hay muchos enunciados que pueden analizarse con este mecanismo, nos vemos en la necesidad de considerar comportamientos excepcionales. Cada excepción es introducida incrementalmente como una regla parcial, sin necesidad de revisar las reglas más generales ni sus condiciones de aplicabilidad. Simplemente,

cada comportamiento excepcional es introducido en relación con el tipo de los objetos que lo requieren. La función de ligadura dinámica se encarga de organizarlos y aplicar el adecuado al ser invocado el módulo semántico.

La aproximación funcional a las reglas que impone el formalismo de reglas genéricas, por otro lado, se adapta perfectamente a nuestro entorno de representación semántica. En general, es el planteamiento más natural para extender sistemas de interpretación semántica categoriales o basados en la aplicación funcional.

Por último, las reglas genéricas nos han proporcionado una interfaz idónea entre sintaxis y semántica. Los fenómenos semánticos son tratados mediante reglas semánticas, sin necesidad de que reglas sintácticas y semánticas estén emparejadas.

Los mecanismos de representación e interpretación semántica que hemos presentado son, en principio, reutilizables en otros dominios. Resuelven problemas generales de procesamiento del lenguaje. El conocimiento específico del dominio viene dado por la información léxica, tomada a su vez de la base de conocimientos. Los requisitos sobre el dominio son: que exista una jerarquía de conceptos, por un lado, y que una representación semántica funcional tenga sentido. Cualquier corpus de problemas científicos tiene estas características. De hecho, basta con introducir nuevos conceptos en la base de conocimiento relativos a un campo como , por ejemplo, la mecánica, para que el sistema esté en condiciones de interpretar problemas de mecánica. Este proceso se describe en [Castells et al., 1993].

Naturalmente, el tipo de interpretación semántica que utiliza el sistema, basado en la lógica, es inadecuado para tratar el lenguaje natural en aquellos dominios en los que no se hace referencia a un conocimiento formal específico como el matemático. En este sentido, nuestro formalismo no es adecuado para sistemas que procesen lenguas naturales sin restricciones.

Respecto a la cobertura del sistema, la interacción de las reglas sintáctica y semántica es capaz de analizar aproximadamente un 65% de los enunciados del corpus. Las causas más comunes para que el sistema no sea capaz de interpretar un enunciado son las siguientes:

- El enunciado contiene fenómenos sintácticos y semánticos complejos que aparecen con muy poca frecuencia, y para los que el esfuerzo de resolver e implementar era desproporcionado frente al número de apariciones en el corpus.
- El enunciado requiere un conocimiento matemático más alla de la información semántica que proporciona la base de conocimientos. La tarea de procesar estos enunciados no puede hacerse de acuerdo con principios generales de parsing, sino a través de soluciones ad-hoc sin sentido en otros dominios.
- La información contenida en las fórmulas es demasiado compleja, y no es

reconciliable con nuestra aproximación basada en la aplicación funcional. La información matemática puede estar mezclada con información sobre formateado. Este es un problema de aceptar como input lenguaje TEX, que es al fin y al cabo un lenguaje de procesamiento de textos y no de representación matemática.

Veamos algunos ejemplos.

```
"Decir en qué punto de la parábola $ y = -x^2+6x-4 $ se verifica
que la tangente a la parábola es perpendicular a la
recta $x + 12y = 0$"

(define ((%p PARABOLA (parabola $ y = -x^2+6x-4 $)))
  (yields
    (pfind
      (pthe (<> %x point)
            (pand (in %x %p)
                  (perpendicular (tangent %p %x)
                                 (line $x + 12y = 0$)))))))
```

Este problema no puede ser resuelto de forma natural mediante una gramática categorial básica: las dependencias a larga distancia, como la de 'qué punto de la parábola', no pueden ser analizadas simplemente mediante aplicación funcional. Otro problema es la segunda aparición de 'la parábola', que hace referencia a la primera. Este sería un problema sencillo si tuviéramos en cuenta fenómenos de discurso, pero está en contradicción con el tratamiento sencillo - y válido en la mayoría de los casos - que hemos dado a los determinantes.

```
"Calcular $x\ge 0$ para que la integral
$ \int_0^x (1-t) \log (t+1) dt$ alcance su valor máximo"

(pfind
  (abs-max
    (realfun (lambda (x)
               (definite-integral $ \int_0^x (1-t) \log (t+1) dt$)))
    (interval :closed 0 :open :infinity)))
```

En este caso, la traducción al lenguaje formal PRÓGENES no tiene una relación composicional directa con la semántica de los elementos de la oración. Se necesita un conocimiento del dominio mayor que la información que extraemos de la base de conocimientos.

Conclusiones

La posibilidad de expresar formalmente reglas gramaticales con una interpretación por defecto tiene varias ventajas, tanto desde el punto de vista del procesamiento como desde la perspectiva de la representación del conocimiento.

Desde el punto de vista de la representación del conocimiento lingüístico, el formalismo de reglas genéricas presenta las siguientes características:

- Permite dar cuenta, de una forma lingüísticamente motivada, de la relación entre el comportamiento regular y excepcional en una gramática. De esta forma se pueden establecer generalizaciones relevantes que no pueden introducirse en una interpretación monótona de las reglas gramaticales.
- Favorece el desarrollo incremental y modular del conocimiento gramatical en los sistemas computacionales, proporcionando a la descripción de relaciones entre objetos lingüísticos una funcionalidad equiparable a la que ofrece la herencia por defecto en la representación del léxico.
- Permite expresar declarativamente, en el nivel gramatical, conocimientos que suelen expresarse como heurísticas asociadas a los procesos de parsing o en niveles metagramaticales.
- Adapta conceptos de programación orientada a objetos, como el de *ligadura dinámica*, a la representación de conocimiento lingüístico.
- El planteamiento funcional de las reglas gramaticales propicia el procesamiento de abajo arriba. Aunque puede argumentarse que este hecho va en detrimento de la declaratividad del sistema, refleja con fidelidad los acercamientos categoriales a la sintaxis y a la interpretación semántica.
- La precedencia entre reglas es deducida por el sistema, no definida externamente. Este hecho simplifica el desarrollo incremental de la gramática. Al escribir las reglas no ha de tenerse en cuenta, a priori, las relaciones jerárquicas entre ellas. Las reglas han de establecerse sobre los tipos de objetos adecuados, y el sistema deduce un orden parcial entre ellas.
- La organización polimórfica de las reglas permite establecer de forma natural una interfaz entre procesos sintácticos y semánticos en la que no es necesario

mantener un emparejamiento entre reglas sintácticas y semánticas. Ambos procesos pueden ser descritos modularmente, interaccionando sólo allí donde sea necesario.

Desde el punto de vista del procesamiento, nuestro formalismo de reglas genéricas ofrece una forma declarativa y modular de reducir el espacio de búsqueda asociado a los procesos de parsing sin necesidad de alterar los algoritmos de parsing introduciendo recetas heurísticas. Además, introduce el comportamiento no monótono de tal manera que la mayoría de las técnicas de parsing desarrolladas para gramáticas libres de contexto son aplicables sobre reglas genéricas. A cambio, introduce un factor de complejidad cuadrático en el tamaño de la jerarquía léxica sobre las que se especifican las reglas de la gramática.

Las técnicas relacionadas con el formalismo descrito han sido usadas y probadas en el entorno del sistema PRÓGENES . Allí hemos comprobado la utilidad de las reglas genéricas para describir procesos de composición semántica a partir de la composición funcional como regla universal por defecto, así como a relacionarlos modularmente con los procesos de búsqueda asociados a la sintaxis.

Respecto al trabajo futuro, existen varias lineas de investigación abiertas. Uno de los campos más prometedores para la aplicación de las reglas genéricas es el de la composición semántica, que ha sido poco desarrollado en esta memoria. En primer lugar, estamos estudiando la interacción entre las aproximaciones categoriales a la interpretación semántica con un conjunto de reglas sintagmáticas, en la línea de [Pereira, 1990]. En segundo lugar, un fenómeno más concreto que se resiste a ser analizado mediante estructuras de rasgos es el de la composición semántica de los nombres compuestos en inglés, como hemos mencionado en la memoria. La forma de establecer y administrar precedencia de las reglas genéricas puede adaptarse muy bien al tipo de fenómenos relacionados con los nombres compuestos [2]. Un trabajo previo sería la implementación del mecanismo de reescritura entre reglas genéricas y - en este caso - reglas gramaticales dentro del formalismo LKB, en el que ya existe implementado un tratamiento de los nombres compuestos mediante una jerarquía de reglas cuyo problema es, precisamente, que no pueden ser interpretadas como reglas por defecto por el sistema.

Dentro del entorno de PRÓGENES , uno de las tareas que estamos acometiendo es la de implementar un módulo de reescritura entre expresiones semánticas globales que permita dar cuenta de aquellos enunciados que se comportan de manera no composicional y específica del dominio. En las reglas de reescritura de este módulo debe jugar un papel fundamental la información asociada a las fórmulas.

[2]Debo esta sugerencia a Ted Briscoe.

Bibliografía

[Ait-Kaci, 1984] Ait-Kaci, H. (1984). *A Lattice-Theoretic Approach to Computation Based on a Calculus of Partially Oredered Types.* PhD thesis, University of Pennsylvania.

[Ajdukiewicz, 1935] Ajdukiewicz, K. (1935). Die syntaktische konnexitat. *Studia Philosophica*, 1.

[Bar-Hillel, 1953] Bar-Hillel, Y. (1953). A quasi arithmetical notation for syntactic description. *Language*, 29.

[Barton Jr. et al., 1987] Barton Jr., G., Berwick, R., and Ristad, E. (1987). *Computational Complexity and Natural Language.* MIT Press.

[Bobrow et al., 1990] Bobrow, D., Demichiel, L., Gabriel, R., Keene, S., Kiczales, G., and Moon, D. (1990). *Common Lisp Object System*, chapter 28. Digital Press.

[Booch, 1990] Booch, G. (1990). *Object Oriented Design With Applications.* Benjamin/Cummings.

[Bos et al., 1994] Bos, J., Mastenbroek, E., McGlashan, S., Millies, S., and Pinkal, M. (1994). The verbmobil semantic formalism. Technical report, Universitat des Saarlandes.

[Bouma, 1992] Bouma, G. (1992). Feature structures and nonmonotonicity. *Computational Linguistics, 18-2.*

[Bresnan, 1982] Bresnan, J., editor (1982). *The Mental Representation of Grammatical Relations.* MIT Press.

[Briscoe, 1991] Briscoe, E. (1991). Lexical issues in natural language processing. In Klein, E. and Veltman, F., editors, *Natural Language and Speech.* Springer-Verlag.

[Calder, 1993] Calder, J. (1993). Some notes on priority union. In Briscoe, E., Copestake, A., and De Paiva, V., editors, *Inheritance, Defaults and the Lexicon.* Cambridge University Press.

[Calder et al., 1988] Calder, J., Klein, E., and Zeevat, H. (1988). Unification categorial grammar: A concise, extendable grammar for natural language processing. In *COLING 88.*

[Cardelli, 1984] Cardelli, L. (1984). A semantics of multiple inheritance. In Kahn, G., McQueen, D., and Plotkin, G., editors, *Semantics of data types.* Springer, N.Y.

[Carpenter, 1992] Carpenter, R. (1992). *The logic of feature structures.* Cambridge University Press.

[Carpenter, 1993] Carpenter, R. (1993). Skeptical and credulous default unification with application to templates and inheritance. In *Inheritance, Defaults and the Lexicon.* Cambridge University Press.

[Carpenter and Pollard, 1991] Carpenter, R. and Pollard, C. (1991). Inclusion, disjointness and choice: The logic of linguistic classification. In *29th annual meeting of the ACL.*

[Carroll, 1994] Carroll, J. (1994). Relating complexity to practical performance in parsing with wide-coverage unification grammars. In *Proceedings of the* 32^{nd} *Annual Meeting of the ACL.*

[Castells, 1994] Castells, P. (1994). *Heuristicas y metaconocimiento en resolucion automatica de problemas.* PhD thesis, Universidad Autonoma de Madrid, Depto. de Ingenieria Informatica.

[Castells et al., 1992a] Castells, P., Diaz, J., Gonzalo, J., Moriyon, R., Rodriguez-Marin, P. Saiz, F., and Tobar, M. (1992a). A scientific problem solver with a natural language interface. In *Seventh International Symposium on Computer and Information Sciences:ISCIS VII.*

[Castells et al., 1992b] Castells, P., Diaz, J., Gonzalo, J., Moriyon, R., Rodriguez-Marin, P. Saiz, F., and Tobar, M. (1992b). Un sistema para la resolucion de problemas enunciados en lenguaje natural. In *Primer Congreso Nacional de Programación Declarativa, PRODE 92.*

[Castells et al., 1993] Castells, P., Moriyon, R., Saiz, F., and Villa, E. (1993). A model of knowledge in elementary mechanics. In *Proceedings of the Internacional Conference on Computers in Education.*

[Chomsky, 1956] Chomsky, N. (1956). Three models for the description of language. *IRE Trans. on Information Theory,* 2(3).

[Chomsky, 1959] Chomsky, N. (1959). On certain formal properties of grammars. *Information and Control,* 2.

[Copestake, 1992] Copestake, A. (1992). The acquilex lkb: Representation issues in semi-automatic acquisition of large lexicons. In *Proceedings of the 3rd conference on Applied Natural Language Processing.*

[Daelemans, 1988] Daelemans, W. (1988). A model of dutch morphophonology and its applications. *AI communications.*

[Daelemans and De Smedt, 1994] Daelemans, W. and De Smedt, K. (1994). Default inheritance in an object-oriented representation of linguistic categories. *International Journal on Human-Computer Studies*, 41.

[Daelemans et al., 1992] Daelemans, W., De Smedt, K., and Gazdar, G. (1992). Inheritance in natural language processing. *Computational Linguistics, vol 18-2.*

[Dahl and McCord, 1983] Dahl, V. and McCord, M. (1983). Treating coordination in logic grammars. *Computational Linguistics*, 9(2).

[Dahl et al., 1995] Dahl, V., Tarau, P., Moreno, L., and Palomar, M. (1995). Treating coordination with datalog grammars. cmp-lg/9505006.

[Díaz and Rodríguez-Marín, 1991] Díaz, J. and Rodríguez-Marín, P. (1991). Prógenes: La interfaz de lenguaje natural. *Boletín de la SEPLN n. 11.*

[De Smedt, 1990] De Smedt, K. (1990). *Incremental Sentence Generation.* PhD thesis, Katholieke Universiteit Nijmegen.

[Diaz, 1995] Diaz, J. (1995). *Un formalismo para la extraccion de informacion semantica en textos matematicos.* PhD thesis, Universidad Autonoma de Madrid, Depto. de Ingenieria Informatica.

[Diaz et al., 1993] Diaz, J., Gonzalo, J., and Rodriguez, P. (1993). El formalismo semantico en la interfaz de lenguaje natural de PRÓGENES ogen. *Boletin de la Sociedad Española para el Procesamiento del Lenguaje Natural*, (14).

[Dowty, 1988] Dowty, D. (1988). Type raising, functional composition, and non-constituent conjunction. In *Categorial Grammars and Natural Language Structures.* Reidel Publishing Company.

[Dörre and Dorna, 1993] Dörre, J. and Dorna, M. (1993). Cuf: a formalism for linguistic knowledge representation. In *Computational aspects of constraint based linguistic descriptions I.* DYANA-2 Deliverable R1.2.A, Universit t Stuttgart.

[Evans and Gazdar, 1989a] Evans, R. and Gazdar, G. (1989a). Inference in datr. In *Proceedings of the Fourth Conference of the European Chapter of the ACL.*

[Evans and Gazdar, 1989b] Evans, R. and Gazdar, G. (1989b). The semantics of datr. In *Proceedings of the Seventh Conference of the Society for the Study of Artificial Intelligence and the simulation of behaviour.*

[Fickinger, 1987] Fickinger, D. (1987). *Lexical rules in the hierarchical lexicon.* PhD thesis, Stanford University.

[Fong, 1985] Fong, S. (1985). New approaches to parsing conjunction using prolog. In *IJCAI 85.*

[Gazdar et al., 1985] Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar.* Blackwell.

[Gazdar and Mellish, 1989] Gazdar, G. and Mellish, C. (1989). *Natural Language Processing in LISP.* Addison-Wesley.

[Gerdemann and King, 1993] Gerdemann, D. and King, J. (1993). Typed feature structures for expressing and computationally implementing feature coocurrence restrictions. In *Proceedings of 4. Fachtagung der der Sektion omputerlinguistik der Deutschen Gessellschaft f ur Sprachwissenschaft.*

[Gonzalo, 1995] Gonzalo, J. (1995). An object-oriented approach to treat exceptions in grammar. In *Grammar formalisms for NLP.* University of Tuebingen Seminar fur Sprachwissenschaft technical report series, to appear.

[Gonzalo et al., 1993] Gonzalo, J., Rodriguez-Marin, P., and Diaz, J. (1993). Formal representation of mathematical texts. *Les chemins du texte.*

[Gonzalo and Solias, 1995] Gonzalo, J. and Solias, T. (1995). Generic rules and non constituent coordination. Enviado al Fourth International Workshop on Parsing Technologies.

[Gotz and Meurers, 1995] Gotz, T. and Meurers, D. (1995). Compiling hpsg type constraints into definite clause programs. In *Proceedings of the 33rd Annual Meeting of the ACL.*

[Jones, 1992] Jones, B. (1992). Predicting nominal compounds. Master's thesis, Clare College, Cambridge University.

[Kartunnen and Kay, 1985] Kartunnen, L. and Kay, M. (1985). Structure sharing with binary trees. In *Proceedings of the 23rd Annual Meeting of the ACL.*

[Kasami, 1965] Kasami, T. (1965). An efficient recognition and syntax algorithm for context-free languages. Technical report, Air Force Cambridge Research Lab., Bedford, Mass.

[Kasper and Rounds, 1986] Kasper, R. and Rounds, W. (1986). A logical semantics for feature structures. In *26th meeting of the ACL.*

[Kasper and Rounds, 1990] Kasper, R. and Rounds, W. (1990). The logic of unification in grammar. *Linguistics and Philosophy,* 13(1).

[Kay, 1983] Kay, M. (1983). Unification grammar. Technical report, Xerox Palo Alto Research Center.

[Kay, 1985] Kay, M. (1985). Parsing in functional unification grammar. In Dowty, D., Karttunnen, L., and Zwicky, A., editors, *Natural Language Parsing.*

[Lambek, 1958] Lambek, J. (1958). The mathematics of sentence structure. *American Mathematical Monthly*, 65.

[Lascarides et al., 1994] Lascarides, A., Briscoe, E., Asher, N., and Copestake, A. (1994). Order independent and persistent typed default unification. Technical report, Centre for Cognitive Research.

[Lavelli and Stock, 1990] Lavelli, A. and Stock, O. (1990). When something is missing: ellipsis, coordination and the chart. In *COLING 90.*

[Maxwell and Kaplan, 1994] Maxwell, J. and Kaplan, R. (1994). The interface between phrasal and functional constraints. *Computational Linguistics*, 19(4).

[Mcgregor, 1990] Mcgregor, R. (1990). Loom user manual. Technical report, USC/ISI.

[Meyer, 1988] Meyer, B. (1988). *Object-Oriented Software Construction.* Prentice Hall.

[Milward, 1990] Milward, D. (1990). Coordination in an axiomatic grammar. In *Coling 90.*

[Milward, 1994] Milward, D. (1994). Non-constituent coordination: theory and practice. In *COLING 94.*

[Moortgat, 1988] Moortgat, M. (1988). *Categorical Investigations: Logical and Linguistic aspects.* Foris, Dordrecht.

[Morrill, 1994] Morrill, G. (1994). *Type Logical Grammar. Categorial Logic of signs.* Kluwer, Dordrecht.

[Morrill and Solias, 1993] Morrill, G. and Solias, T. (1993). Tuples, discontinuity and gapping in categorial grammar. In *EACL-93.*

[Pereira, 1990] Pereira, F. (1990). Categorial semantics and scoping. *Computational Linguistics 16-1.*

[Pereira and Warren, 1980] Pereira, F. and Warren, D. (1980). Definite clause grammars for natural language analysis - a survey of the formalism and a comparison with augmented transition networks. *Artificial Intelligence*, (13):231–278.

[Pereira and Warren, 1983] Pereira, F. and Warren, D. (1983). Parsing as deduction. In *21st annual meeting of the association for computational linguistics.*

[Pollard and Sag, 1987] Pollard, C. and Sag, I. (1987). *Information-based syntax and semantics. Volume 1: Syntax.* CSLI Lecture Notes 13. Chicago UNiversity Press.

[Pollard and Sag, 1994] Pollard, C. and Sag, I. (1994). *Head-Driven Phrase Structure Grammar.* University of Chicago Press and CSLI publications.

[Pulman, 1989] Pulman, S. (1989). Unification and the new grammatism. In *Theoretical Issues in Natural Language Processing.* Yorick Wilks.

[Ross, 1988] Ross, J. (1988). *Constraints on variables in Syntax.* PhD thesis, MIT.

[Saiz, 1994] Saiz, F. (1994). *Fundamentacion de un sistema para la resolucion automatica de problemas.* PhD thesis, Universidad Autonoma de Madrid, Depto. de Ingenieria Informatica.

[Selman and Levesque, 1989] Selman, B. and Levesque, H. (1989). The tractability of path-based inheritance. In *11th IJCAI.*

[Shieber, 1986] Shieber, S. (1986). *An introduction to unification-based approaches to Grammar.* CSLI.

[Shieber, 1992] Shieber, S. (1992). *Constraint-Based Grammar Formalisms. Parsing and Type Inference for Natural and Computer Languages.* MIT Press.

[Shieber et al., 1994] Shieber, S., Schabes, Y., and Pereira, F. (1994). Principles and implementation of deductive parsing. Technical Report TR-11-94, Center for Research in Computing Technology, Harvard University.

[Shieber et al., 1983] Shieber, S., Uskoreitz, H., Pereira, F., Robinson, J., and Tyson, M. (1983). The formalism and implementation of patr-ii. In *Research on interactive acquisition and use of knowledge.* Artificial Intelligence Center, SRI International.

[Solias, 1992] Solias, M. (1992). *Gramáticas Categoriales, Coordinación Generalizada y Elisión.* PhD thesis, Universidad Autonoma de Madrid.

[Solias, 1993] Solias, T. (1993). Sequence product, gapping and multiple wrapping. In *Proceedings of the workshop on Linear Logic and Lambek Calculus.*

[Sowa, 1984] Sowa, J. (1984). *Conceptual Structures: Information Processing in Mind and Machine.* Addisson-Wesley.

[Stroustrup, 1992] Stroustrup, B. (1992). *The C++ programming language, second edition.* Addison-Wesley.

[Touretzky, 1986] Touretzky, D. (1986). *The mathematics of inheritance systems.* Morgan-Kaufmann.

[van den Berg and Prust, 1991] van den Berg, M. and Prust, H. (1991). Common denominators and default unification. In *Proceedings of the Computational Linguistics in the Netherlands, first meeting.*

[Woods-73, 1973] Woods-73 (1973). An experimental parsing system for transition network grammars. In Rustin, R., editor, *Natural Language Processing.* Algorithmics Press, New York.

[Younger, 1967] Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control 10.*

[Zajac, 1992] Zajac, R. (1992). Inheritance and constraint-based grammar formalisms. *Computational Linguistics 18,2.*

[Zeevat and Calder, 1987] Zeevat, H. Klein, E. and Calder, J. (1987). An introduction to unification categorial grammar. In *Edinburgh Working Papers in Cognitive Science, vol. 1: Categorial Grammar, Unification Grammar and Parsing.*